Proceedings of Machine Learning Research vol vvv:1–14, 2024

LaDyBot: Learning Language-Guided Collaborative Dynamics

Xihang Yu XIHANGYU@UMICH.EDU
College of Literature, Science, and the Arts, University of Michigan, Ann Arbor, MI, USA

Elizabeth A. Olson LIZOLSON@UMICH.EDU
Robotics Department, University of Michigan, Ann Arbor, MI, USA

Odest Chadwicke Jenkins OCJ@UMICH.EDU
Robotics Department, University of Michigan, Ann Arbor, MI, USA

Abstract

In order to successfully execute manipulation tasks under human guidance, robotic agents need to interpret language-directed commands. Understanding these instructions provides robots with crucial semantic insights that enhance their ability to comprehend scenes. This paper concentrates on the specific challenge of manipulating objects in motion, guided by verbal instructions. While this task poses significant challenges for robotic systems, it is relatively straightforward for humans. Our innate ability to rapidly discern and categorize various motion patterns enables us to adapt our movements accordingly. The integration of language instructions into the understanding of dynamic environments remains largely unexplored. This paper delves into this issue through two distinct case studies. Our first case study centers on a table-top manipulation task involving moving objects in an industrial context, such as executing commands like "*Grasp the objects that are pushed linearly*" Here, language instructions are utilized not only to pinpoint the targeted object but also to enhance the prediction of its future positions. By developing a feedback controller and a motion planner that leverages these predicted poses, we have achieved an almost perfect success rate in simulations. The significance of human-robot interaction is also underscored in our studies. An illustrative example is a scenario where a human requests a service robot to "*Toast our wine glasses*" This interaction highlights the importance of understanding and responding to human language cues in the development of assistive robotic technologies. In our second task, we shift our focus to recognizing hand actions and predicting hand poses guided by language instructions, as explored in Garcia-Hernando et al. (2018). This approach enhances outcomes compared to methods that do not incorporate language guidance. Across both tasks, we investigate a solution involving open-vocabulary language instructions. We introduce the concept of an "*instruction bag*"—a collection of diverse instructions tailored to describe a specific task accurately. Our findings demonstrate that the application of this instruction bag significantly boosts performance.

Keywords: language-guided manipulation, learning dynamic, feedback control

1. Introduction

Human-robot collaboration hinges on the robot's ability to perceive and understand language. By processing language instructions, robots gain two key advantages: firstly, they can focus more effectively on specific targets and objects; secondly, they acquire additional semantic insights that enhance scene comprehension, including understanding the motion models of objects and their interrelationships. Conversely, in the realms of human-robot interaction and collaboration, grasping the dynamics of human movement and object manipulation is crucial. This is because dynamic behavior is a common aspect of the real world. Humans have honed their skills to adapt to dynamic

© 2024 X. Yu, E.A. Olson & O.C. Jenkins.

"Raise our glasses for celebration!"
"How can my experience of toasting inform the execution of this task?"

Figure 1: A figure illustrating toast wine task via language command.

environments in various scenarios, ranging from catching a volleyball to chasing rabbits, or from handing over objects to throwing them. However, previous research in language-guided manipulation has predominantly focused on static environments. Recognizing this gap, our work aims to extend these principles to dynamic settings, where objects are in dynamic motion.

This leads us to pose a critical question: *Is it possible to develop a system capable of interpreting random human-like language instructions, with the dual aims of (i) learning the dynamics of humans and objects, and (ii) predicting their future dynamics?* In our research, we explore the integration of language into the learning of sequential and dynamic information. This is achieved by employing a pretrained language model, where language acts as a means to encapsulate complex dynamic phenomena. Our project introduces an object-centric framework. It leverages language descriptions of tasks and the state information of objects to predict the future dynamics of the objects in question. This approach represents a significant step forward in the realm of human-robot interaction, emphasizing the power of language in enhancing robotic comprehension and predictive capabilities.

Contributions. Our contributions are two folds:

- A language-guided machine learning system capable of classifying various types of motion and accurately predicting future dynamics.
- We have conducted extensive experimental validation, utilizing both simulated moving objects and a real-world hand-action dataset, to demonstrate the efficacy of our approach.

Paper organization. This paper is structured in the following manner: Section 2 provides an overview of the relevant literature. The problem formulation is presented in Section 3. Detailed discussions on the algorithm's realization and implementation are found in Section 4, which is followed by an in-depth analysis of our experiments in Section 5 and 6. Conclusion will be presented in Section 7.

2. Related Works

2.1. Language-Conditional Manipulation

Language-conditioned manipulation is a widely studied area. Existing vision language manipulation (VLM) algorithms Shridhar et al. (2021, 2022); Zheng et al. (2022); Liu et al. (2022b,a); Jiang et al. (2023); Driess et al. (2023); Lynch et al. (2023) have demonstrated impressive performance on object manipulation tasks. While CLIPort proposed by Shridhar et al. (2021) only deals with 3 DoF desktop object rearrangement tasks, it has been extended to work with 6-DoF manipulation Zheng et al. (2022). Additionally, StructFormer proposed in Liu et al. (2022b) uses object-centric model in object rearrangement (i.e. directly predicting the relative pose of objects) and has good results on the specific shape formation tasks. StructDiffusion Liu et al. (2022a) improves the performance of StructFormer by incorporating a diffusion model in the pipeline. However, StructFormer and StructDiffusion only focus on object rearrangements. In addition, Liu et al. (2022b) and Liu et al. (2022a) use pre-segmented point clouds which are often not accessible for robotics applications. More recently, Perceiver-Actor proposed by Shridhar et al. (2022), has achieved record-breaking performance in general language-conditional tasks. Jiang et al. (2023) focuses on a multi-modality input mixed language with image prompts. Driess et al. (2023) deals with manipulation and navigation tasks by outputting end-to-end action. Lynch et al. (2023) also deals with manipulation, with emphasis on language feedback. All of these works solving language-conditional tasks are limited to static environments. Lynch et al. (2023) contains task to move objects but does not learn the dynamic patterns of objects explicitly. Our work will generalize language-based inference to dynamic grasping tasks.

2.2. Dynamic Manipulation

In the field of dynamic manipulation, Zeng et al. (2020) learns residual velocity to compensate the uncertainty of a physics-based controller (a known dynamic model like gravitational field). Eisner et al. (2022) predicts a predefined vector flow for articulation objects from vision and selects the grasp point based on the vector flow of highest magnitude. To grasp moving objects, a closed-loop visual feedback controller for dynamic environment was implemented in Morrison et al. (2020). After the best grasp pose has been predicted, a velocity command is computed to make the gripper reach the target pose. Song et al. (2020) also formulates dynamic grasp as a closed-loop policy generation problem. In contrast to Morrison et al. (2020), Song et al. (2020) formulates action policy generation as a Markov Decision Process. An object-centric approach by Akinola et al. (2021) uses RNN to predict future object pose sequence from a sequence of instantaneous poses. Wu et al. (2022) formulates the dynamic problem as a "move-and-grasp" game and uses adversarial reinforcement learning to train the grasping policy and object moving strategies jointly. Most recent work Jia et al. (2023b) deals with dynamic grasping in obstacle-cluttered environment by merging look ahead time and time budget into the framework. However, all existing methods do not include language inferences in the training process.

3. Representation and Algorithm

3.1. Representation

Our research addresses the challenge of classifying object dynamics into predefined action classes, denoted as $\mathcal{C}$. Consider, for instance, a table-top scenario where object dynamics include linear and circular motions. In the context of hand actions, "high five" and "toast wines" represent two distinct action classes. We define the space of poses as $\mathcal{P}$. For the table-top object manipulation task, the pose space is represented as $P = \mathbb{R}^{1\times 3}$, reflecting the fact that we are considering single rigid bodies. In contrast, for the hand action prediction task, the pose space is denoted as $P = \mathbb{R}^{21\times 3}$, corresponding to the 3D locations of the 21 joints in a hand model.

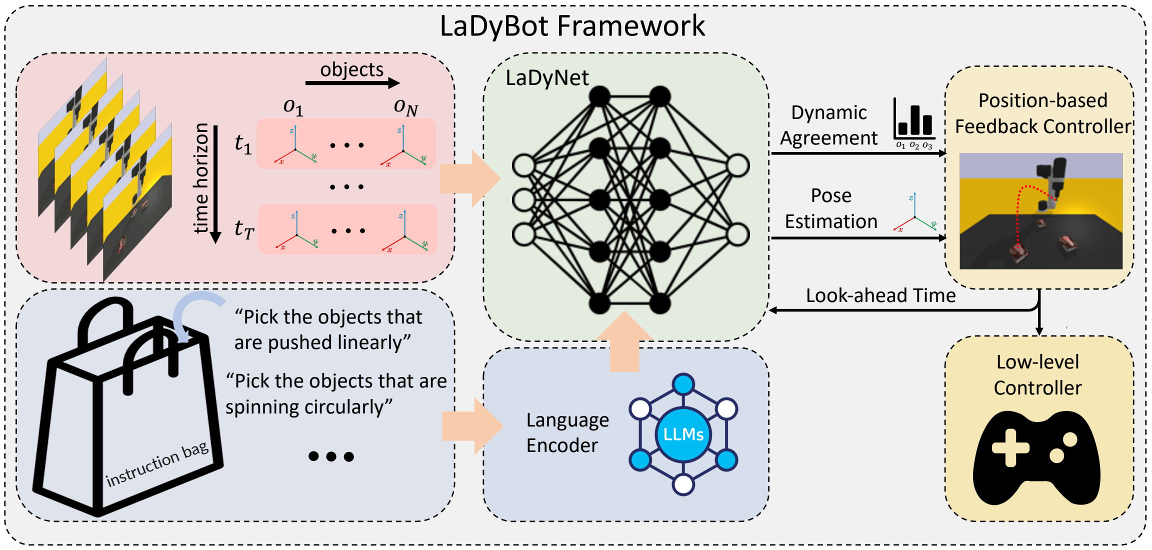

Figure 2: We introduce the LaDyBot framework, specifically tailored for table-top grasping tasks, which uniquely integrates language embeddings. The framework operates by concatenating the poses of all objects $o_1, \ldots, o_N$ within a time window of size W, ranging from t_k to t_{k-W+1}. This is combined with the look-ahead time $t_{k+L} - t_k$ and language instruction l. LaDyNet processes these inputs and identifies the target object whose dynamics align with the given language description, also predicting the pose of this object at the future time t_{k+L}. Subsequently, a position-feedback controller determines the next position for the end effector, as well as a new look-ahead time L. Finally, a low-level controller is responsible for ensuring the precise convergence of the end effector to the desired position.

3.2. Algorithm

Figure 2 provides an illustrative overview of our algorithm. Consider a video sequence of T frames, each frame featuring N objects. Associated with each object o_i is a language instruction l_i, where $i \in \{1, \ldots, N\}$, describing its dynamics. This instruction l_i comprises a set of word tokens $w_1, \ldots, w_{m_i}$, with m_i representing the number of tokens in l_i, elucidating the dynamics of the specified object in the frame. The frames undergo preprocessing to extract the poses $p_{i,k}$ for each object o_i across the frames $t_k, k \in \{1, \ldots, T\}$. At any given frame t_k, our focus is on a window of W preceding frames, spanning from t_{k-W+1} to t_k. Given the poses $p_{i,k-W+1}, \ldots, p_{i,k}$ for all objects $\{o_1, \ldots, o_N\}$ and a lookahead frame L from the current frame, our objectives are twofold: (1) to classify the dynamics $c_i \in \mathcal{C}$ for each object o_i, and (2) to predict the future pose $p_{i,k+L}$ of object o_i at time t_{k+L}. In the experiments section, we incorporate the language instructions $l_i, i \in \{1, \ldots, N\}$ into our inputs. Let $P_{i,k,W}$ denote the set $\{p_{i,k-W+1}, \ldots, p_{i,k}\}$ for each object. Specifically:

$$c_1, ..., c_N = \mathcal{F}_c \left(P_{1,k,W}, ..., P_{N,k,W}, L | l_1, ..., l_N\right), \forall i \tag{1}$$

$$p_{1,k+L}, ..., p_{N,k+L} = \mathcal{F}_p \left(P_{1,k,W}, ..., P_{N,k,W}, L | l_1, ..., l_N\right), \forall i \tag{2}$$

where $\mathcal{F}_c$ and $\mathcal{F}_p$ are dynamics classification and pose prediction networks respevtively.

4. Implementation

This section is structured for ease of understanding as follows: The design of the network is detailed in Section 4.1. This is followed by an exposition of the proposed instruction bag in Section 4.2. Lastly, the intricacies of the feedback controller are presented in Section 4.3.

4.1. Network Design

Our focus is on determining the action class of an object and its future pose, denoted as $p_{\text{lookahead}}$, based on the previous poses of several objects within a given time window and a language instruction. The LaDyNet network, designed for this task, comprises a fully connected architecture with two hidden layers, each consisting of 256 units. The final hidden layer connects to two separate output layers, dedicated to dynamics prediction and pose estimation, respectively. The effectiveness of this network is evaluated through a specially designed loss function. This function is a weighted combination of Mean Squared Error (MSE) for pose estimation accuracy and Cross Entropy (CE) for object prediction reliability:

$$L_{\text{MSE}} = \frac{1}{B}\sum_{q=1}^{B}(\hat{Y}_{\text{pose},q} - Y_{\text{pose},q})^2 \tag{3}$$

$$L_{\text{CE}} = \frac{1}{B}\sum_{q=1}^{B}\sum_{j=1}^{|C|} Y_{\text{select},q,j}\log(\hat{Y}_{\text{select},q,j}) \tag{4}$$

$$L = \lambda L_{\text{MSE}} + L_{\text{CE}} \tag{5}$$

where B is the batch size, $|C|$ is the number of classes, λ is a weight parameter, $Y_{\text{pose},q}$ is the ground truth pose for the q-th sample in the batch, $\hat{Y}_{\text{pose},q}$ is the estimated pose for the q-th sample in the batch, $Y_{\text{select},q,j}$ is the ground truth label for the j-th class of the q-th sample in the batch, $\hat{Y}_{\text{select},q,j}$ is the predicted probability for the j-th class of the q-th sample in the batch.

4.2. Instruction Bag

Drawing inspiration from the 'bag of words' concept in Natural Language Processing, we introduce the concept of an 'instruction bag.' Initially, each canonical instruction is processed through ChatGPT to generate n_l semantically similar instructions. These n_l instructions are then divided into two parts. The first n_1 sentences, along with the canonical instruction, form the instruction bag. During training, we randomly select from this instruction bag for each data point, resulting in $n_1 + 1$ pairs of linear and circular instructions in the training instruction bag. The remaining $n_2 = n_l - n_1$ sentences are used to create the testing dataset, referred to as noncanonical instructions. For instance, in the table-top grasping task, the phrase dataset is detailed in Table 1. Here, two motion primitives—linear and circular—are each represented by a canonical instruction: 'Pick the objects that are pushed linearly' and 'Pick the objects that are spinning circularly', respectively. These instructions are input into ChatGPT, yielding 13 sentences for each. Ultimately, this process provides us with 10 sentences for the training dataset and 4 for the testing dataset for each motion primitive.

Circular	**Linear**
training dataset	
Pick the objects that are spinning circularly	Pick the objects that are pushed linearly
Grab objects swirling in a perfect circle	Grab objects zooming in a straight line
Grab items that are twirling in a loop	Grab objects that are sliding in a straight path
Identify objects that are orbiting in a curve	Identify objects that are sliding straight ahead
Select objects that are whirling in a circular route	Select objects that are darting in a straight line
Choose items tracing a roundabout trajectory	Choose items zipping along a ruler-straight path
Snag objects whizzing in a round motion	Snag objects that are being thrust in a linear direction
Go for the things that are looping round and round	Go for the things that are moving dead straight
Pick the items that are spinning like a top	Pick items that are just shooting straight ahead
Snatch up anything that's moving in a circle	Snatch up anything that's cruising in a straight path
test dataset	
Identify objects moving in loops	Identify objects that are proceeding directly forward
Choose objects that are engaged in a rotating motion	Choose objects that are progressing in a straight path
Select objects that are pushed in a circular motion	Select objects that are pushed along a linear trajectory
Determine objects displaying a rotational motion	Determine objects displaying a straightforward motion

Table 1: Comparison of phrases used in training and test datasets for Circular and Linear motion.

4.3. Feedback Controller

Utilizing the previously described network, we classify object actions and predict future poses in conjunction with language instructions. However, robotic tasks often require coordination with dynamically moving target objects. For instance, a robot may need to grasp a moving object, high-five a moving human hand, or toast with a moving wine glass. To address these dynamic scenarios, we propose a coordination controller that enables the robot to effectively converge with the target objects. This is implemented through a position-based feedback controller, designed to operate at frame t_k with a specified number of look-ahead frames L. The controller functions as follows:

$$\begin{aligned} t_{k+L} &= \alpha\|p_{\mathrm{ee},k} - p_{d,k}\| + t_k \qquad (6) \\ p_{\mathrm{ee},k+1} &= p_{\mathrm{ee},k} + \min\left(1, \frac{\beta}{t_{k+L}-t_k}\right) \cdot (p_{d,k+L} - p_{\mathrm{ee},k}). \qquad (7) \end{aligned}$$

where $p_{\mathrm{ee},k}$ represents the pose of the end effector at frame k, and $p_{d,k}$ is the predicted pose of the target object at frame k, where $d \in \{1, \dots, N\}$ denotes the target object. Our controller calculates a look-ahead time t_{k+L} based on the current positions of the object, $p_{d,k}$, and the end effector, $p_{\mathrm{ee},k}$, with α as a key hyperparameter. The underlying principle of this approach is that a greater distance between the end effector and the object necessitates a longer look-ahead time, ensuring sufficient time for the end effector to converge with the object. The second equation in our model operates under the premise that both the end effector and the object should move towards the predicted look-ahead position $p_{d,k+L}$, as forecasted by $\mathcal{F}_p$, the output of the LaDyNet pose prediction model. Meanwhile, $\mathcal{F}_c$ functions implicitly within the pipeline, classifying the dynamics of all objects before predicting the specific target object $d \in \{1, \dots, N\}$. This controller is applied iteratively across all time frames $t_i, i = 1, \dots, T$.

5. Tabletop Grasping Simulation

5.1. Setup

5.1.1. Dataset Generation

In our study, we have established a dynamic grasping environment within the PyBullet simulator. This setup features a UR5 robotic arm equipped with a Suction end effector. The objects intended for manipulation are positioned on a continuously moving conveyor belt. To simulate various object behaviors, we have devised two principal categories of trajectories: linear and circular. These trajectories can be efficiently randomized by adjusting a specific set of parameters. The following section provides a detailed description of how we simulate these object trajectories:

- Line. Firstly we sample a starting pose and direction in the manipulation space. Then random sample velocities from $U(0.3, 0.4)$. To create more balanced training dataset, we use stractified sampling strategy for velocities. Stractified sampling strategy is that we divide the velocity range into n_s equally subranges. When sampling, first sample a subrange and then uniformly sample velocity in the subrange. In the experiment, we set $n_s = 10$.
- Circle. Firstly we sample a circle center in the manipulation space. Then random sample radius from $U(0.01, 0.05)$ and angular velocity from $U(10, 11)$. For angular velocities, we use stractified sampling. Fianlly, the direction of circular motion will be sampled, either moving clock-wise or counter clock-wise.

In each episode of dynamic grasping, we generate 3 distinct trajectories for 3 pudding boxes, sourced from the YCB dataset Calli et al. (2017). Our dataset comprises 60 linear and 60 circular episodes, with each episode capturing 400 poses. The time interval between consecutive poses is set at $\frac{1}{240}$ seconds. We adhere to a train/test split of 4/1. For data processing, every set of 50 consecutive poses is grouped as a single data point. Notably, all poses are relative to their preceding pose, a strategy that enhances the pose predictor's ability to generalize across different trajectory locations. This training methodology aligns with the approach used in Jia et al. (2023a). Each data point in our study is a composite of a 50-pose moving window serving as the pose input, paired with a future timestamp designated as the look-ahead time.

5.1.2. Training Profile

Throughout the training process, we randomly select a language instruction from the instruction bag for each data point. The language instruction, denoted as l, is preprocessed using the RoBERTa pretrained language model, serving as our language encoder. We employ only the last hidden layer from RoBERTa's sequential output, as it fully encapsulates sentence-level information. The input of our network is concatenated inputs of object poses, language instructions, and look-ahead time. The output of the network includes both the predicted object of interest and the pose of this object at the look-ahead time, relative to the current timeframe. Our network operates with a batch size of 512 and utilizes the Adam optimizer, set to a learning rate of 0.0001. The training extends over 200 epochs, with the loss function assigned a weight of $\lambda = 50.0$. It's important to note that we currently do not employ a pose estimator for object poses. Instead, we use ground truth poses and introduce Gaussian noise, $w_p \sim N(0, 0.01)$, to each 3D pose to simulate estimated poses. This approach aims to mirror real-world conditions more closely. As part of our future plans, we intend to integrate vision-based pose estimation into our pipeline for a more comprehensive and effective system.

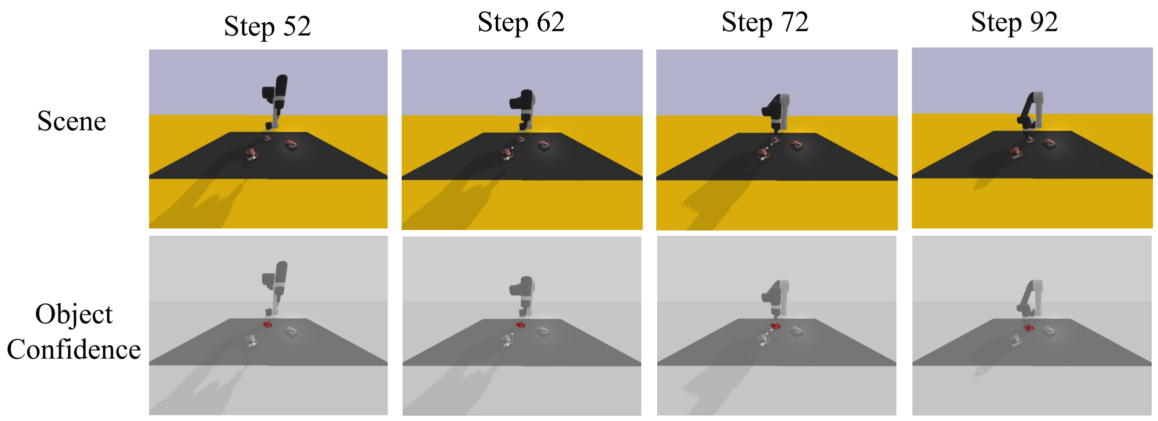

Figure 3: This illustration depicts a typical sequence within a grasping task, where the objective is to pick up an object that is being pushed linearly. The sequence showcases scenes at steps 52, 62, 72, and the final step. Within these scenes, there are three objects present, but only one is selected as the target, highlighted in red in the second row of images. In the first row, the future positions of the chosen object are indicated by white balls. As the sequence progresses, these white balls gradually align more closely with the target object. This alignment culminates in step 92, where the end effector successfully grasps the target. In the second row, the confidence levels in object prediction are visualized through a heat map. Notably, the target object (marked in red) consistently exhibits significantly higher confidence scores compared to the other objects, often nearing a probability of 1. This clear distinction in confidence levels underlines the effectiveness of our prediction model in accurately identifying and tracking the target object throughout the task.

5.2. Baseline

Given the novelty of learning language-guided dynamics, we established our own baseline for comparison. In this baseline model, the network is trained without utilizing the instruction bag for sampling. To ensure a fair comparison, we maintained consistency in the training profile and dataset. The sole distinction lies in the use of canonical instructions throughout the training. These instructions are "Pick the objects that are pushed linearly" and "Pick the objects that are spinning circularly." For this baseline training, the network follows the same parameters as our primary model, operating with a batch size of 512, across 200 epochs, and employing a loss weight of $\lambda = 50.0$.

5.3. Performance Analysis

5.3.1. LADYBOT

The object prediction success rate and pose prediction error in Test dataset are shown in Table 2. Pose Prediction Error e is defined as:

$$e = \frac{1}{D} \sum_{q=1}^{D} \|\hat{Y}_{pose,q} - Y_{pose,q}\| \tag{8}$$

where D is the number of data points in testing dataset and $Y_{pose,q}$ and $\hat{Y}_{pose,q}$ are ground truth pose and estimated pose respetively. The first column shows that canonical instructions are used in

Table 2: LaDyBot trained with instruction bag. Object Prediction Success Rate ↑/ Pose Prediction Error (m) ↓.

	Canonical Instruction	NI 1	NI 2	NI 3	NI 4
Linear	0.9583/0.0429	0.9578/0.0453	0.9351/0.0501	0.9221/0.0541	0.9065/0.0607
Circular	0.9659/0.0386	0.4565/0.0953	0.9336/0.0475	0.3804/0.1040	0.2122/0.1155

Table 3: LaDyBot trained with instruction bag. Grasping Success Rate ↑ / Wrong Pick Rate ↓ / Fail Pick Rate ↓. GT poses: Ground truth poses. LB: LaDyBot. NI i: Noncanonical instruction i.

	GT poses	LB	LB+Low Noise	LB+High Noise
Linear	1.0/0.0/0.0	0.9/0.0/0.1	0.9/0.1/0.0	0.9/0.1/0.0
Circular	1.0/0.0/0.0	1.0/0.0/0.0	1.0/0.0/0.0	1.0/0.0/0.0
	LB+NI1	LB+NI2	LB+NI3	LB+NI4
Linear	1.0/0.0/0.0	0.9/0.0/0.1	1.0/0.0/0.0	1.0/0.0/0.0
Circular	0.9/0.1/0.0	1.0/0.0/0.0	0.5/0.3/0.2	0.3/0.7/0.0

testing. The other four columns show the results if using four noncanonical instrustions in testing. The order number of noncanonical instructions is the same as top-down order in Table 1. Except for three noncanonical instructions for circular motion, all others achieved close to 1 object prediction success rate while average pose prediction error remains around 5 cm. In simulation, we run 10 unseen linear sequences and 10 unseen circular sequences in PyBullet simulator. Table 3 shows the simulation results. For GT poses experiments, we know the target object and ground truth $p_{d,k+L}$ at any specific time t_k with look ahead time t_{k+L}. We can see that GT poses achieve 100% success rates for both linear and circular sequences. This suggests that our position-based feedback controller works quite well given that estimated $p_{d,k+L}$ is accurate. Low Noise means Gaussian Noise $N(0, 0.001m)$ is added to 3D poses. High Noise means Gaussian Noise $N(0, 0.01m)$ is added to 3D poses. Wrong Pick refers successful grasping but wrong pick of a distracted object. Fail Pick refers to fail to pick any objects due to inaccurate future pose estimation. All sequences achieved 1.0 or close to 1.0 grasping success rate including high noise sequences except for two noncanonical instructions case for circular motion. Note that the failure of the two noncanonical circular instructions is mainly due to false selection as shown in Table 2 which suggests that the unseen language instruction is the main reason for defeats. A typical sequence of using LaDyBot is shown in Fig. 3.

5.3.2. No instruction bag

We perform an ablation study on instruction bag to study how the performance of language generalization changes. In Table 4, the object prediction success rate and pose prediction error in Test dataset are shown. We can see that the use of instruction bag improves prediction success rate for noncanonical instructions while reducing the pose prediction errors by a large margin. Only exceptions are circular motion for NI 1 and NI 4. The simulation results exhibit similar trend which are shown in Table 5. We can see that using instruction bag improves the grasping success rates for almost all instructions while the use of instruction bad does not affect the success rate for noisy pose estimation. Within the failure grasping cases, both wrong selection and wrong pose estimation lead to the failure, which may differ across different instructions. In summary, the use of instruction bag improves the robustness of the whole system.

Table 4: LaDyBot trained without instruction bag. Object Prediction Success Rate ↑/ Pose Prediction Error (m) ↓.

	Canonical Instruction	NI 1	NI 2	NI 3	NI 4
Linear	$0.9628/0.0422$	$0.5152/0.1437$	$0.5486/0.1345$	$0.8049/0.0871$	$0.8384/0.0842$
Circular	$0.9624/0.0375$	$0.4757/0.0847$	$0.9161/0.0488$	$0.2370/0.1190$	$0.4470/0.0967$

Table 5: LaDyBot trained without instruction bag. Grasping Success Rate ↑ / Wrong Pick Rate ↓ / Fail Pick Rate ↓. GT poses: Ground truth poses. LB: LaDyBot. NI i: Noncanonical instruction i.

	LB	LB+Low Noise	LB+High Noise	
Linear	$0.9/0.1/0.0$	$1.0/0.0/0.0$	$0.8/0.0/0.2$	
Circular	$0.9/0.0/0.1$	$1.0/0.0/0.0$	$0.9/0.0/0.1$	
	LB+NI1	LB+NI2	LB+NI3	LB+NI4
Linear	$0.3/0.4/0.3$	$0.2/0.8/0.0$	$0.7/0.0/0.3$	$0.6/0.3/0.1$
Circular	$0.1/0.3/0.6$	$1.0/0.0/0.0$	$0.0/0.6/0.4$	$0.2/0.4/0.4$

6. Hand Action Experiments

In our second task, we shift focus to language-guided hand action recognition and hand pose prediction. For effective human-robot collaboration, it's crucial for a robot to understand both current human actions and anticipate future intentions. This makes hand action recognition and hand pose prediction vital components of human-robot interaction. Our subsequent experiments leverage the F-PHAB hand action dataset Garcia-Hernando et al. (2018). This dataset includes 1,175 action videos across 45 different action categories, contributed by 6 subjects. Each video sequence is labeled with its corresponding action, and every frame within these sequences is annotated with hand pose information. These annotations were obtained using a motion capture system that automatically determines the 3D world frame locations of each of the 21 joints in a hand model. This is achieved through the use of 6 magnetic sensors and inverse kinematics. The next section details our framework's approach to hand action recognition and 63D hand pose prediction, considering a specified look-ahead time.

6.1. Setup

In this experiment, our goal is to determine the hand dynamics classification $c_i \in \mathcal{C}$ for each hand sequence i and predict the future hand pose $p \in \mathcal{P}$ at a specified look-ahead time t. We focus on classifying 'high five' and 'toast wine' actions, while also predicting the hand pose at a future moment within these sequences. This experiment utilizes the first-person hand pose dataset referenced in Garcia-Hernando et al. (2018), which provides annotations for previous poses. In real-world settings, magnetic sensors and inverse-kinematics can be used to capture hand pose data. Regarding language instructions, 9 variants were generated by ChatGPT based on canonical instructions like "Raise our glasses in celebration" for toasting wine, and "Slap me a high five" for high fives. We compiled an instruction bag with 7 sentences, combining 6 prompted instructions with one canonical instruction for each action. The training instructions are detailed in Table 6, and 3 noncanonical instructions are reserved for testing.

The network architecture mirrors that of the previous section, though the input dimension differs due to the complex nature of hand poses. Inputs to the network include a window of past hand

High Five	**Toast Wine**
training dataset	
Slap me a high five	Let's clink our glasses
Let's do a high five	Raise your glass for a cheer
Hit me with a high five	Time to celebrate with a drink
Up for a high five	Join me in a drink toast
High five me!	Let's celebrate with a glass raise
Let's slap a five	Raise our glasses in celebration
Give me a friendly high five	Here's to a wonderful moment
test dataset	
High five for the good times	Cheers to good times
Let's smack those hands high	Toast to our gathering
High five for our success	Celebrate this moment with a toast

Table 6: Comparison of phrases used in training and test datasets for High Five and Toast Wine actions.

Table 7: LaDyBot trained **with** and **without** instruction bag in the first table. LaDyBot trained without using language in the second table. Action recognition success rate $\uparrow$ / Pose Prediction Error (m) $\downarrow$.

	Canonical Instruction	Noncanonical Instruction
w instruction bag	1.0000/0.4437	0.9609/0.4660
w/o instruction bag	1.0000/0.4446	0.7204/0.5775
not using language	0.9476/0.4403	

poses p over $\frac{2}{3}$ seconds, a look-ahead time t, and a language instruction l. The language instruction is processed using a pretrained RoBERTa model. Our training utilizes PyTorch with the Adam optimizer. We adopt a leave-one-out protocol for the dataset comprising 6 subjects, training on five and reserving one for testing. This approach ensures network adaptability to unseen subjects. The choice of 'high five' and 'toast wine' actions is twofold: they are commonplace in social interactions, and they necessitate the robot's ability to interpret human language and predict hand poses during interaction. The training parameters are set as follows: loss weight $\lambda = 20.0$, learning rate 0.00001, batch size 512, and the network is trained over 100 epochs.

6.2. Performance Analysis

6.2.1. LaDyBot

The outcomes of our experiments are presented in Table 7, with the pose prediction error defined as per Equation 8. The results indicate a remarkable achievement in the classification task, boasting a 100% success rate. Even when utilizing noncanonical instructions, the classification success rate remains impressively high at 96%. In terms of pose estimation, we observed an average error of approximately 0.44m per joint when using canonical instructions, and a slight increase to 0.46m with noncanonical instructions. A representative sequence of hand action prediction is visually detailed in Fig. 4.

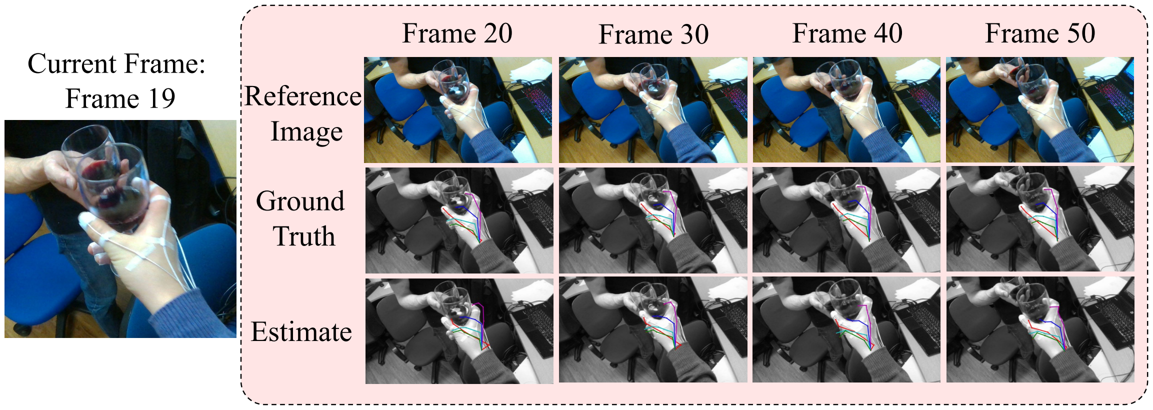

Figure 4: Illustration of Hand Action Prediction. Language instruction and hand poses from frame 0 to frame 19 are input into the system. LaDyBot classifies the hand action as "toast wine" and predict the hand poses in the future frame 20, 30, 40, 50 respectively.

6.2.2. Ablation Study

This section delves into an ablation study focusing on the use of the instruction bag. A comparative analysis of the performance with and without the instruction bag is detailed in Table 7. Notably, the incorporation of the instruction bag results in enhanced pose estimation for canonical instructions and a significant improvement in both the classification success rate and pose estimation accuracy for noncanonical instructions. Furthermore, we conducted an ablation study on the role of language in the process. In this variant, the network's inputs were limited to poses and the look-ahead time, excluding language instructions. The outcomes of this modification are also presented in Table 7. While there was a marginal decrease in pose estimation error—dropping to 0.4403 from 0.4437 for canonical instructions and to 0.4660 for noncanonical instructions—compared to the LaDyBot using the instruction bag, a notable impact was observed in hand action classification. The success rate for this task was lower than that achieved by LaDyBot, even with noncanonical instructions (0.9609 vs. 0.9476). These results underscore the significant role that language instructions play in enhancing the understanding of hand actions.

7. Conclusion

This paper showcases the profound effect of embedding language commands into robotic systems for dynamic environments. Our exploration, encompassing tabletop manipulation and hand action recognition, leverages an 'instruction bag' strategy, yielding marked enhancements in both robotic efficacy and environmental comprehension. These advances underscore the vital importance of language in elevating human-robot interaction, setting the stage for robots that are more adaptive and intuitively responsive. Looking ahead, our future endeavors will focus on experimenting with various sequential transformer-based networks and extending our research into real-world applications.

Acknowledgments

I extend my heartfelt thanks to Liz and Chad for their exceptional guidance and mentorship. I am also grateful to Boyang for the insightful discussions regarding machine learning algorithms.

References

Iretiayo Akinola, Jingxi Xu, Shuran Song, and Peter K Allen. Dynamic grasping with reachability and motion awareness. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9422–9429. IEEE, 2021.

Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Ben Eisner, Harry Zhang, and David Held. Flowbot3d: Learning 3d articulation flow to manipulate articulated objects. *arXiv preprint arXiv:2205.04382*, 2022.

Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 409–419, 2018.

Yinsen Jia, Jingxi Xu, Dinesh Jayaraman, and Shuran Song. Learning a meta-controller for dynamic grasping. *arXiv preprint arXiv:2302.08463*, 2023a.

Yinsen Jia, Jingxi Xu, Dinesh Jayaraman, and Shuran Song. Learning a meta-controller for dynamic grasping. 2023b.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: Robot manipulation with multimodal prompts. 2023.

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

Weiyu Liu, Chris Paxton, Tucker Hermans, and Dieter Fox. Structformer: Learning spatial structure for language-guided semantic rearrangement of novel objects. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6322–6329. IEEE, 2022b.

Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023.

Douglas Morrison, Peter Corke, and Jürgen Leitner. Learning robust, real-time, reactive robotic grasping. *The International journal of robotics research*, 39(2-3):183–201, 2020.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3):4978–4985, 2020.

Tianhao Wu, Fangwei Zhong, Yiran Geng, Hongchen Wang, Yongjian Zhu, Yizhou Wang, and Hao Dong. Grasparl: Dynamic grasping via adversarial reinforcement learning. *arXiv preprint arXiv:2203.02119*, 2022.

Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4): 1307–1319, 2020.

Kaizhi Zheng, Xiaotong Chen, Odest Chadwicke Jenkins, and Xin Eric Wang. Vlmbench: A compositional benchmark for vision-and-language manipulation. *arXiv preprint arXiv:2206.08522*, 2022.