# Natural Language Guided Goals for Robotic Manipulation

*Andrew Nichols Crawford Taylor*
*University of Utah*

UUCS-24-007

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

29 April 2024

## *Abstract*

Goal specification for robotic manipulation is a critical area of research, defining how we interact with robots and how we communicate with them. We propose using universally guided diffusion to generate natural language goal aligned rigid body transforms, from segmented point clouds. We explore various methods for guidance, including their challenges and strengths, and discuss which future directions would be fruitful. This thesis explores endowing robots with the ability to understand and fulfill human goals, potentially enhancing their utility and versatility in various domains.

# NATURAL LANGUAGE GUIDED GOALS FOR ROBOTIC MANIPULATION

by

Andrew Nichols Crawford Taylor

A Senior Honors Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the

Honors Degree in Bachelor of Science

In

Computer Science

Approved:

[illegible]
Alan Kuntz
Thesis Faculty Supervisor

Mary Hall
Mary Hall
Director, Kahlert School of Computing

Thomas C. Henderson
Thomas C. Henderson
Honors Faculty Advisor

Monisha Pasupathi, PhD
Dean, Honors College

April 2024

## ABSTRACT

Goal specification for robotic manipulation is a critical area of research, defining how we interact with robots and how we communicate with them. We propose using universally guided diffusion to generate natural language goal aligned rigid body transforms, from segmented point clouds. We explore various methods for guidance, including their challenges and strengths, and discuss which future directions would be fruitful. This thesis explores endowing robots with the ability to understand and fulfill human goals, potentially enhancing their utility and versatility in various domains.

## TABLE OF CONTENTS

# 1 - INTRODUCTION

Robots are commonly used to automate difficult or repetitive tasks. Current robots are commonly used in industrial manufacturing, and are beginning to transition towards more general tasks and task conditions. For most industrial robots, conditions are engineered to be within a small range. The robot's motion is repetitive, often programmed without awareness of its environment. In newer applications for robotics, such as food service or warehousing, the task conditions may vary, but the goal remains consistent. Learning-based methods can help adapt to these uncertainties and variable conditions. In such cases, the task goal remains consistent and could be programmed as a desired state of known objects, a photo depicting the desired state, or a structured description of the relationships between objects.

For robots to move towards tasks that vary both in their conditions and purpose specifying those tasks must be intuitive and simple. Programming motions or high-level task goals would render the robot inaccessible and less useful for many, particularly for simple tasks. For example, for many people fetching an object or putting away groceries would be easier to perform than to program a computational description for. Natural language instructions, were robots able to understand them, hold promise as a modality to enable intuitive user interactions with robots, moving them toward general purpose assistants.

People with disabilities, people with chronic conditions, or elderly people could benefit especially from these home assistants. Home tasks for many can be exceptionally difficult, and robotic home assistants could enable people like this to live more independently or easily, and to age in place. With traditional approaches that require goal state specification

[1] or more time intensive or complex interactions with the robot, people requiring care may not find robotic assistants useful. However being able to interact with natural language commands makes robots intuitive.

In this work, we build on Universally Guided Diffusion [2] and StructDiffusion [3] to explore approaches for natural language aligned goals using Universal Guidance for Diffusion. We explore the difficulties and various approaches for guiding diffusion on rigid object transformations.

# 2 - BACKGROUND

## 2.1 - DEEP LEARNING

Deep learning is a fundamental component of current robotics research. It grants roboticists the ability to approximate arbitrary functions given input-output pairs. While there are many variations being studied, they share many commonalities. We create functions whose output is determined by their input, and a set of weights, $\theta$. For example:

$$f_\theta(\mathbf{x}) = \mathbf{\Theta}\mathbf{x} = \mathbf{y} \tag{2.1}$$

This function has a vector of inputs who combine to an output vector, where the weights (a matrix of size (output dimension, input dimension)) can be adjusted to approximate a linear function, but not non-linear functions [4]. This is of limited use in this form, since nearly all the functions we are interested in approximating are not mathematically linear. To account for this, we add a non-linearity function $g(\cdot)$ and a bias term $\mathbf{b}$.

$$f_\theta(\mathbf{x}) = g(\mathbf{\Theta x} + \mathbf{b}) = \mathbf{y} \tag{2.2}$$

This is still limited, since it is just a biased weighted average with introduced non-linearity. To further its ability to approximate, we can add additional layers, each with its own weight matrix (and weight matrix size), constrained by its output size needing to be equal to the input size of the next layer.

$$f_\theta(\mathbf{x}) = h_\theta(h_\theta(h_\theta(...(\mathbf{x})))) = \mathbf{y}, \text{ and } h_\theta(\mathbf{x_n}) = g(\mathbf{\Theta x_n} + \mathbf{b}) = \mathbf{x_{n+1}} \tag{2.3}$$

While this in theory can replicate any other function, manually adjusting each weight in practice is impractical. Acquiring the weights to the values that approximate the function we want is difficult. To do this, we create an optimization problem. We construct a function $L(\cdot)$ (our loss function) that decreases the better we are approximating, and minimize it.

$$\underset{\theta}{\operatorname{argmin}}\, L(f_\theta(\mathbf{x})) \tag{2.4}$$

To do this, we use gradient descent. Since we can calculate the derivative of the loss function with respect to the weights, we can evaluate it for each input/output pair, and adjust each weight such that the loss is reduced. We adjust proportional to a learning rate, and repeat this adjustment to minimize the loss.

$$\theta \coloneqq \theta - \alpha \nabla_\theta L(f_\theta(\mathbf{x})) \tag{2.5}$$

This is done for some set of inputs, $\mathbf{x}$, true outputs, $\mathbf{y}$, and predicted (approximated) outputs, $\hat{\mathbf{y}}$. L for example could be the mean squared error between the true and predicted outputs, $|\hat{\mathbf{y}} - \mathbf{y}|^2$.

We use the chain rule to calculate those derivatives, propagating backwards from the loss term. We calculate the change in the loss for each component of the layer, and then use that to compute the previous layer, and so on.

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{x_n}} &= \frac{\partial L}{\partial \mathbf{x_{n+1}}} \cdot \frac{\partial \mathbf{x_{n+1}}}{\partial \mathbf{x_n}} \\
\frac{\partial L}{\partial \mathbf{\Theta}} &= \frac{\partial L}{\partial \mathbf{x_n}} \cdot \frac{\partial \mathbf{x_n}}{\partial \mathbf{\Theta}} \\
\frac{\partial L}{\partial \mathbf{b}} &= \frac{\partial L}{\partial \mathbf{x_n}} \cdot \frac{\partial \mathbf{x_n}}{\partial \mathbf{b}}
\end{aligned} \tag{2.6}$$

Thus we can adjust each weight in each layer towards an optimal solution. When constructing such a network and optimizing it, there are several important practical considerations. First, we must choose the number of layers, and the size of each weight matrix. These are two important hyper-parameters. Broadly speaking, the more parameters we have the more expressive the function can be. In addition, we have to choose our learning rate, $\alpha$, for each step. One common method is Adam [5], which uses momentum (analogous to a ball with momentum rolling down a hill) to help avoid local minima.

When optimizing this function, we must keep in mind that our data is not complete. We have access only to a set of true examples of the function, and if we optimize for their performance exclusively, we will overfit to the data. Thus it is a fundamental practice of deep learning to separate data used for the optimization process (training) from data used to evaluate whether the optimization process is working (validation). This attempts to ensure that we are fitting appropriately to the true function.

There are further practical considerations as well. To further avoid overfitting, we randomly disable a proportion of the weights, a technique called dropout [6]. The choice of non-linearity, or activation, function and weight initialization can impact training dramatically [7], where the restricted linear unit, ReLU (equivalent to $\max(0, x)$), or its variants are the most common choice. Further Layer Normalization [8] increase the stability and ease of training by reducing domain shift within the network. He et al. [9] introduced residual connections, inspired by the idea that it might be easier to learn how to adjust the input than it would be to reconstruct all of the information contained in the input, reformulating layers as $h_\theta(g(h_\theta(x))) + x$.

Other improvements come from not just the training process and practical techniques, but the structure of the network. Additionally, by altering the structure of the network, we can use it for different purposes, such as data generation. This can also help us improve accuracy.

### 2.1.1 - STRUCTURES

The structure described in the previous section, a Multi-Layer-Perceptron or MLP, is still widely used as a component part in larger networks, but in many applications has been superseded by structures that take advantage of the symmetry in the data for more efficient learning.

A popular structure, called a Convolutional Neural Network, or CNN, takes advantage of the spatial nature of many applications. Often applied to images, (or 3d voxel data), it uses a learned kernel to create outputs invariant to translations in image space. For example, an edge would give the same high output wherever it was recognized within the image. This is great for data-efficiency. The intuition for this is that if we learn to recognize a cat as a pattern within an image, we don't actually need to see cats in every possible position. We only need to learn the one kernel, and it will generalize over the translational symmetry.

With this and many other structures, backpropagation is adapted to the structure following the same method. The only difference is the change of derivative – which in practice is computed automatically.

Another important aspect of these structures to consider is the equivariance. Similar to invariance but instead of being indifferent to changes in input, the output experiences a corresponding change

$$\begin{aligned} \text{Equivariance: } f(g(x)) &= g'(y) \\ \text{Invariance: } f(g(x)) &= y \end{aligned} \tag{2.7}$$

where $f(x)$ is the function we are describing, and $g$ and $g'$ are the symmetric operation and its corresponding symmetric operation in the output space. While many times we may want invariance (a rotated cat is still a cat), equivariance is also extremely important. For example in robotics if our state is rotated, for example a different object facing a different direction, we want to grab it the same but with a corresponding rotation in our grasp. Building these symmetries into our networks helps us not have to learn from every rotation (or other symmetry), and thus reduces the amount of data and training we need.

Much further along in the history of deep learning, another architecture became common. The Attention Mechanism keeps many of the benefits of past approaches, and further improves parallelization and scaling on common GPU architectures. Attention operates on sequences of vectors - again in a residual fashion as in [9]. When thinking about attention as an operation we can view it as distributing and accumulating information about the vectors it operates on.

For each of the vectors, we generate a query and key vector from a query and key matrix. These can be viewed as what information that vector might be interested in, and what information the value might provide. These vectors are in the same space, so if we take a dot product between each pair, we find the relevance of each key to each query. Using these relevancy scores, we can compute a change in the input vector from a weighted sum of each key and its relevance and a value matrix which translates the key information into our desired update. This is captured by the formula

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.8}$$

where $d_k$ is the dimension of keys $K$.

Self Attention, where the key vectors and query vectors are generated from the same source, was popularized in an architecture called the Transformer [10] and has been shown to be remarkably effective [11].

Diffusion, a more recent technique, isn't a network structure but a process. We can leverage structures for a more efficient diffusion process, but what's special about it is how we train and use the network. We'll discuss diffusion further in Section 2.3.

## 2.2 - ROBOTIC MANIPULATION APPROACHES

Learning-based systems for robotic manipulation can be categorized into several broad categories. The robot must perceive, understand, and act to achieve a goal. Perhaps the simplest approach is to map robot inputs to robot outputs. This function we call a policy $\pi$, where the robot outputs an action for any given state. These states and actions can have many different forms. The states can be observations such as camera inputs, or global and exact knowledge of object's orientations and positions.

Another approach is inspired by classical approaches to manipulation, planning based approaches search through a space of actions to find sequences that achieve their goals. These often model the world's state, and then compute or predict state transitions for each action. These predictions are used to create a plan.

### 2.2.1 - POLICY LEARNING

Policy learning focuses on directly mapping the robot's current state to its next action. This approach involves learning a policy function, $\pi(s) = a$, where $s$ represents the state (e.g., images, sensor readings, estimated object positions, etc.) and $a$ represents the action to be taken (e.g., push object, move gripper to position, set joint angle, etc.).

Several methods fall under policy learning, such as imitation learning, where the robot learns from demonstrations provided by a human expert, mapping state-action pairs to learn the desired behavior.

With reinforcement learning, (RL), the robot learns through trial and error, receiving rewards for achieving goals and penalties for failures. This allows the robot to adapt to changing environments and learn complex behaviors. Typically in RL, the reward functions must be crafted by a human expert.

Inverse Reinforcement Learning, (IRL), aims to learn the reward function that explains an expert's demonstrated behavior. This reward function can then be used for reinforcement learning, allowing the robot to learn similar behaviors.

### 2.2.2 - PLANNING

Planning-based approaches involve searching for a sequence of actions that achieve a specific goal. These methods often model the robot's state and environment, and then explore possible action sequences to find the best one.

Task and Motion Planning, or TAMP, is a broad category that integrates both task planning (high-level goals) and motion planning (low-level actions) to achieve complex objectives. Search-based planning methods explore a search space of possible action sequences, using graph-based or sampling algorithms like A* or RRT* to find the best path to the goal. Hierarchical Planning breaks down complex tasks into smaller sub-tasks, allowing for more efficient planning and execution.

### 2.2.3 - GOALS

An initial approach to enable a robot to understand a natural language command, for example "place the groceries on the shelf," is to directly train a model to predict the next action or sequence of actions [12][13][14][15][16][17].

Another approach is generating goals such as in [3][18]. Diffusion [19][20] notably is well suited for modeling multi-modal distributions such as the distribution of goals that satisfy a planning task, or realistic poses of objects conditioned on a scene. Unlike directly predicting action sequences generating goals maintains diversity of solutions to any given planning task, and diffusion in particular achieves better results [3][21].

While these techniques show promising results, one main limitation is the requirement to learn a whole language. Robotics as a domain is limited by the cost of dataset creation, and our resulting small dataset sizes make capturing a whole language difficult.

Our proposed contribution is to integrate a model that has already learned the language on a broader scale dataset, and use Universal Guidance for Diffusion Models [2] to guide a diffusion process, trained on realistic poses for objects in the scene, towards a natural language aligned and realistic goal.

### 2.2.4 - LARGE LANGUAGE MODELS IN ROBOTICS

Past work integrating large language models (LLMs) into robotic systems has demonstrated the ability to take natural language instructions and generate plans directly. SayCan, Socratic Models, Inner Monologue, and Code as Policies all interpret natural language into some executable action for robots.

LLM Planning [22] fine tunes an LLM to map known skills to generated language. SayCan [23] uses labeled affordance functions to align natural language planning with robot affordances and corresponding skills. It sequentially selects skills using this method. Socratic Models [24] similarly generates plans with few-shot LLM function call generation. It uses language conditioned policies (specifically CLIPort) to further help interpret the goal. Robustness is improved by dialogue (hence the Socratic portion of the name) between VLMs and the LLMs (and though not demonstrated specifically in robotics, ALMs). Inner Monologue [25] augments SayCan to include feedback, from Socratic models. This enables it to detect and react to failure, or new environmental information. It is also able to qualify goals by asking questions. Code as Policies [26] interprets language goals into code of a high level API for a robot. This can include language conditioned actions, as well as any other. ProgPrompt [27] generates code directly, interpreting natural language as high level discrete planning steps.

Another approach is use LLMs to interpret natural language into structured languages or forms that describe problems, as in PDDL [28], that can be solved with classical planning techniques [29][30][31].

## 2.3 - DIFFUSION

### 2.3.1 - MATHEMATICAL FRAMEWORK

Diffusion [32][33] is conceptualized as a forward and reverse process, where we gradually add noise to a sample until it becomes indistinguishable from random noise, and a reverse process where we gradually remove the noise to regain a clean sample.

- Our gradual process takes the form of a fixed number of time steps, $T$, for each process.
- The noise at each step may have a unique scale, represented by a list $\{\alpha_t\}_{t=1}^{T}$.
- A clean data point is represented with $\mathbf{z_0}$, while a data point with $t$ steps of noise added is represented with $\mathbf{z_t}$.

However, we don't actually add noise incrementally. Since the product of two Gaussians is yet another Gaussian, we can just use the noise schedule to ensure the noise is incrementally increasing and perform only one equivalent step. So we can aquire any timestep's noised sample with

$$\mathbf{z_t} = \sqrt{\alpha_t}\mathbf{z_0} + (\sqrt{1-\alpha_t})\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.9}$$

thus the reverse process with full information can be computed using the same sampled epsilon

$$\mathbf{z_0} = \frac{\mathbf{z_t} - (\sqrt{1-\alpha_t})\boldsymbol{\epsilon}}{\sqrt{\alpha_t}}. \tag{2.10}$$

### 2.3.2 - LEARNED MODEL

To learn this reverse process, instead of predicting directly the next sample in the reverse process with $\mathbf{z_{t-1}} = \epsilon(\mathbf{z_t}, t)$, we learn to predict the noise in the current sample with a network $\epsilon_\theta$

$$\epsilon_\theta(\mathbf{z_t}, t) = \hat{\boldsymbol{\epsilon}} \tag{2.11}$$

and generate the previous time step's sample with a sampling method $\mathbf{z_{t-1}} = S(\mathbf{z_t}, \hat{\boldsymbol{\epsilon}}, t)$. For example,

$$\hat{\mathbf{z_0}} = \frac{\mathbf{z_t} - \sqrt{1-\alpha_t}\hat{\boldsymbol{\epsilon}}}{\sqrt{\alpha_t}} \tag{2.12}$$

and

$$\hat{\mathbf{z_{t-1}}} = \sqrt{\alpha_t}\hat{\mathbf{z_0}} + (\sqrt{1-\alpha_t})\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.13}$$

### 2.3.3 - TRAINING THE MODEL

We train the model on every combination of data points and time steps, $(\mathbf{z_0}, t) \in Z \times T$ and across Gaussian samples $\boldsymbol{\epsilon}$.

For each data point we compute $\mathbf{z_t}$ as described above, and use that sampled $\boldsymbol{\epsilon}$ as our target. Thus a MSE loss is

$$L(\mathbf{z_0}, t) = (\epsilon_\theta(\mathbf{z_t}, t) - \boldsymbol{\epsilon})^2, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{2.14}$$

## 2.4 - UNIVERSAL GUIDANCE FOR DIFFUSION

The universal guidance algorithm covers three augmentations to the diffusion process.

### 2.4.1 - FORWARD UNIVERSAL GUIDANCE

This augmentation modifies the network by augmenting our noise prediction in each sampling step. We augment our noise predictor by adding a term

$$\hat{\epsilon_\theta}(\mathbf{z_t}, t) = \epsilon_\theta(\mathbf{z_t}, t) + s(t) \cdot \nabla_{\mathbf{z_t}} \ell(c, f(\hat{\mathbf{z_0}})) \tag{2.15}$$

that is the gradient $\nabla_{\mathbf{z_t}} \ell(\cdot, \cdot)$ of the closeness $\ell(c, \hat{c})$ between the label $c$, and predicted clean data point $f(\hat{\mathbf{z_0}})$, scaled by a per-sampling step guidance strength $s(t)$.

### 2.4.2 - BACKWARD UNIVERSAL GUIDANCE

This forward guidance is often insufficient because experimentally it can not quite strike a balance between bringing the data point into the real distribution, and also into the aligned distribution. To do this, we compute a change in clean data space

$$\Delta \mathbf{z_0} = \underset{\Delta}{\operatorname{argmin}}\, \ell(c, f(\hat{\mathbf{z_0}} + \Delta)) \tag{2.16}$$

by solving Equation 2.16 with $m$-step gradient descent. Similar to Equation 2.15 we can use this as an augmentation to our noise prediction creating $\tilde{\boldsymbol{\epsilon}}$,

$$\tilde{\boldsymbol{\epsilon}} = \epsilon_\theta(\mathbf{z_t}, t) - \sqrt{\alpha_t/(1-\alpha_t)}\Delta\mathbf{z_0}. \tag{2.17}$$

Notably, the difference between the backward and forward guidance is that the forward guidance is computed in the time step's noised configuration space, and backwards guidance is computed in predicted clean data space, and then converted into the noise space.

### 2.4.3 - PER-STEP SELF-RECURRENCE

Experimentally however, these again prove to be insufficient. To help improve this further, we explore the data manifold further at one noise scale/time step in the diffusion process. This exploration helps us find a sample that both satisfies the desire for realness of our diffusion process, and an accurate match with our guidance function. To do this, we sample $\mathbf{z_{t-1}}$ using our guidance methods, and then apply a regular forward diffusion step with $\boldsymbol{\epsilon}' \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to our guided sample to obtain $\mathbf{z_t}'$. We do this k times before continuing to step $t-1$.

$$\mathbf{z_t}' = \sqrt{\alpha_t/\alpha_{t-1}} \cdot \mathbf{z_{t-1}} + \sqrt{1-\alpha_t/\alpha_{t-1}} \cdot \boldsymbol{\epsilon}'. \tag{2.18}$$

### 2.4.4 - COMBINING THE AUGMENTATIONS

So the combined process is to, for each timescale:

1. Calculate predicted clean datapoint $\hat{\mathbf{z_0}}$. (Eq. 2.12)

2. Create an initial noise prediction using forward universal guidance $\hat{\epsilon_\theta}$. (Eq. 2.15)

3. Modify our noise prediction $\hat{\epsilon_\theta}$ using $m$ steps of gradient descent. (Eq. 2.16)

4. Repeat the above $k$ times per step by sampling to obtain $\mathbf{z_{t-1}}$ then re-noising to obtain $\mathbf{z_t}'$. (Eq. 2.18)

# 3 - METHODS

## 3.1 - PROBLEM DEFINITION

We restrict the domain for this project to single views of a scene of a shelf, our environment, and objects on it. We perceive the scene's point cloud, and assume we're able to segment it into each object's segmented point cloud, $\mathbf{O}_i, i = 1, 2, 3, \ldots, N$. Each point cloud is a subset of the scene's point cloud. We are also given a language instruction, $l$, for which we hope to find a realistic, aligned set of transforms, $\{\mathbf{T}_i\}_{i=1}^{K}, K \leq N$, that would put any needed object in a stable location that accomplishes the goal. So, we hope to find and re-arrange some subset of objects such that they are stable, and aligned with the given goal.

In the framework of Universal Guidance, we hope to minimize the alignment function, $\ell(l, \{\mathbf{T}_i\}_{i=1}^{K})$, meaning a set of transforms most aligned with our language instruction.

$$\underset{\{\mathbf{T}_i\}_{i=1}^{K}}{\operatorname{argmin}} \, \ell(l, \{\mathbf{T}_i\}_{i=1}^{K}) \text{ s.t. } \{\mathbf{T}_i\}_{i=1}^{K} \text{ provide realistic stable transforms} \tag{3.1}$$

The selection and implementation of the alignment function is complex. We'll explore options and their potential benefits and limitations in Section 3.3. To meet our stability constraint, we train a diffusion model to generate poses.

## 3.2 - POSE DIFFUSION

Diffusion has been applied to the poses of objects for the purpose of object rearrangement [3][34]. Similarly, we hope to generate object-wise transforms from a generative process.

Within the diffusion framework, our diffusion process will operate on these transforms. We're also interested in generating poses conditional on the environment and objects visible. We want the diffusion process to place objects on the environment, so we train our noise prediction function to be conditional on the point clouds of the environment and each object.

$$\epsilon_\theta(\mathbf{O}_i, \mathbf{z}_t, t) = \hat{\epsilon} \tag{3.2}$$

We hope that the noise prediction learns broadly the shape of the object and how these shapes can be placed on various environments. To implement this, internal to the noise prediction the point clouds are encoded using network structures that operate on point clouds, such as PointNet [35], PointNet++ [36], PointConv [37], Point Transformer [38], Point Transformer V2 [39], or PointConvFormer [40].

Pose diffusion uses a transformer internally. Each vector that is input to the transformer represents an object or the environment, and our hope is a transformer's symmetries would lead to especially effective learning on this domain. Each object may, for example, be updated with the new de-noised position of each other object to avoid collisions. Thus for each object we create a vector that includes the pose information, time step encoding, and the object encoding.

Since we are diverging from the typical application of transformers it is important to alter several common practices. Typical transformers use a positional encoding with the time step and other properties, but since our sequence of objects is unordered we don't include this. We also adjust the typical attention masking, since we're predicting all future poses at once as opposed to the next one in the sequence. We want them to adjust their positions relative to all other objects, not just objects earlier in the sequence.

Poses, also, have many different representations. We use a 9 dimensional representation specified in past work which is shown to have better performance [41].

## 3.3 - GUIDANCE

The selection and implementation of a guidance function requires careful consideration. What we hope to achieve is to take advantage of large internet-scale models that have learned models of language, and how it corresponds with object locations. With that, perhaps the most obvious choice is a Vision Language Model (VLM), a model that relates image information to text with similarity scores.

### 3.3.1 - VISION LANGUAGE MODELS

Perhaps the most commonly referenced VLM is CLIP [42]. CLIP uses a shared embedding space for images and language, where the embedding is trained on image caption pairs. Similarity is computed with a dot product between the normalized embedding vectors, to find the cosign angle between them.

$$\cos(\theta) = \frac{\mathbf{u}}{||\mathbf{u}||} \cdot \frac{\mathbf{v}}{||\mathbf{v}||} \tag{3.3}$$

This is used as a similarity score, where 1 is most similar, and -1 is most dissimilar. To create our guidance function, we can map to this embedding space by training an additional network on our states.

$$||\psi_{\text{CLIP}}(I) - \psi_{\theta}(\{\mathbf{T}_i\}_{i=1}^{K})||^2 \tag{3.4}$$

Using an image of our transformations in the scene, we train to map directly to the embedding space with an MSE loss, where $\psi_{\text{CLIP}}$ is CLIP's image embedding network, and $\psi_{\theta}$ is our learned function that operates on transforms.

We can then use this network to compute similarity scores between language and our transforms.

$$\ell(l, \{\mathbf{T}_i\}_{i=1}^{K}) = \frac{\psi_{\theta}(\{\mathbf{T}_i\}_{i=1}^{K}) \cdot \phi_{\text{CLIP}}(l)}{||\psi_{\theta}(\{\mathbf{T}_i\}_{i=1}^{K})|| \cdot ||\phi_{\text{CLIP}}(l)||} \tag{3.5}$$

Where $\phi_{\text{CLIP}}$ is CLIP's language embedding network.

### 3.3.2 - LANGUAGE-TO-LANGUAGE

Another attempt at a guidance function would be a text to text similarity model, such as Sentence-BERT [43]. Based on Bidirectional Encoder Representations from Transformers (BERT) [44], a larger scale pre-trained model, it embeds sentences to a shared embedding space we can again train to match, if we generate text at training time that describes the scene. Thus, we can train a function to match:

$$||\psi_{\text{Sentence-BERT}}(S) - \psi_{\theta}(\{\mathbf{T}_i\}_{i=1}^{K})||^2 \tag{3.6}$$

using a sentence descriptor of our scene, $S$. We train to map directly to the embedding space with an MSE loss, where $\psi_{\text{Sentence-BERT}}$ is Sentence-BERT's sentence embedding network, and $\psi_\theta$ is our learned function that operates on transforms.

We can then use this network to compute similarity scores between language and our transforms as follows.

$$\ell(l, \{\mathbf{T}_i\}_{i=1}^K) = \frac{\psi_\theta(\{\mathbf{T}_i\}_{i=1}^K) \cdot \psi_{\text{Sentence-BERT}}(l)}{||\psi_\theta(\{\mathbf{T}_i\}_{i=1}^K)|| \cdot ||\psi_{\text{Sentence-BERT}}(l)||} \tag{3.7}$$

### 3.3.3 - RELATIONAL GUIDANCE

Relations offer a more explicit, but potentially more limited form of guidance. They have been used in past work to specify goals without pose [45] [46]. Relations for a goal can be defined as a goal conjunction, g, of relations

$$\mathbf{g} = r_1 \wedge r_2 \wedge \ldots \wedge r_k, r_i \in \mathcal{R} \tag{3.8}$$

where $\mathcal{R}$ is the set of all possible relations. Relations, in this work, are pairwise between objects. For example, "Object 1 is above Object 2." To achieve our natural language target, we can use a Large Language Model (LLM) [11][47] to generate a set of relations from some examples in a prompt, and the natural language goal.

This avenue is more promising in terms of guidance, as these relations can be hardcoded from the perspective of the robot, where each object being left, right, above, etc. can be calculated and used for guidance efficiently. For each relation then, we define a function that decreases to zero where the relation function is satisfied. For example, if $x_1$ is Object 1's horizontal position and $x_2$ is Object 2's horizontal position we can compute a guidance function with:

$$f(x_1, x_2) = \max(0, e^{x_1 - x_2} - 1). \tag{3.9}$$

This function could be used for the relation “Object 1 left of Object 2,” as it gives a function that when minimized ensures the leftness of Object 1. A combination of these functions could be used to ensure that the constraints implied by a natural language goal are met. Broadly our loss function can be constructed by adding a series of these functions

$$\ell(l, \{\mathbf{T}_i\}_{i=1}^{K}) = f_1(\{\mathbf{T}_i\}_{i=1}^{K}) + f_2(\{\mathbf{T}_i\}_{i=1}^{K}) + \ldots + f_{|g|}(\{\mathbf{T}_i\}_{i=1}^{K}) \tag{3.10}$$

where the functions are ultimately determined by the language instruction.

# 4 - RESULTS

## 4.1 - POSE DIFFUSION

For experimental evaluation of pose diffusion, we first created simulation data to train on. Using four manually-scaled randomly-selected shelves from Objaverse [48], we extracted approximately upright surfaces from the mesh. We adjusted and removed surfaces that objects could not be placed on. We also took a selection of the YCB Objects [49], computed stable placement poses, and curated their upright and forward facing positions. Using Isaac Sim, we placed the objects according to their potential stable poses and the corresponding bounding boxes within the shelves’ upright surfaces. We designed our data generation procedure to limit object velocity and rotational velocity, such that if any objects interpenetrated they would be moved away from each other by the underlying physics engine in a controlled manner. Using this process, we generated 15,259 training samples, and 4,801 validation samples.

We then trained our transformer based diffusion network on this data, and qualitatively evaluated its results on our training environment objects. To evaluate, we examined a set of 64 diffused generations. We conclude that pose diffusion functions well, and we extend past results to show generations that are conditional on environment objects. A selection of diffused results is shown in Figure 4.1.

We can see that in these results, stability is learned from the dataset, with few minor unstable poses. We see several objects in front facing upright positions, and in computed stable poses with random rotations around the normal of the surface. These rotations for each object were programmed into the dataset, and this confirms our ability to learn multi-modal distributions of poses.

Experimentally, while more recent papers suggest that methods such as PointConvFormer may improve results and speed, we experienced implementation issues. Perhaps for larger models this would create an improvement, as suggested by the authors.

## 4.2 - GUIDANCE

### 4.2.1 - VISION LANGUAGE MODELS

To experiment with guidance functions, we train a mapping from our poses to the embedding space of our chosen method, as described in Section 3.3.1. We use the image pose pairs generated as part of our pose diffusion dataset to train the mapping, and our validation data to ensure our closeness score functions well. We search the space of guidance hyper-parameters for good quality matching examples with a grid search, and by manual search.

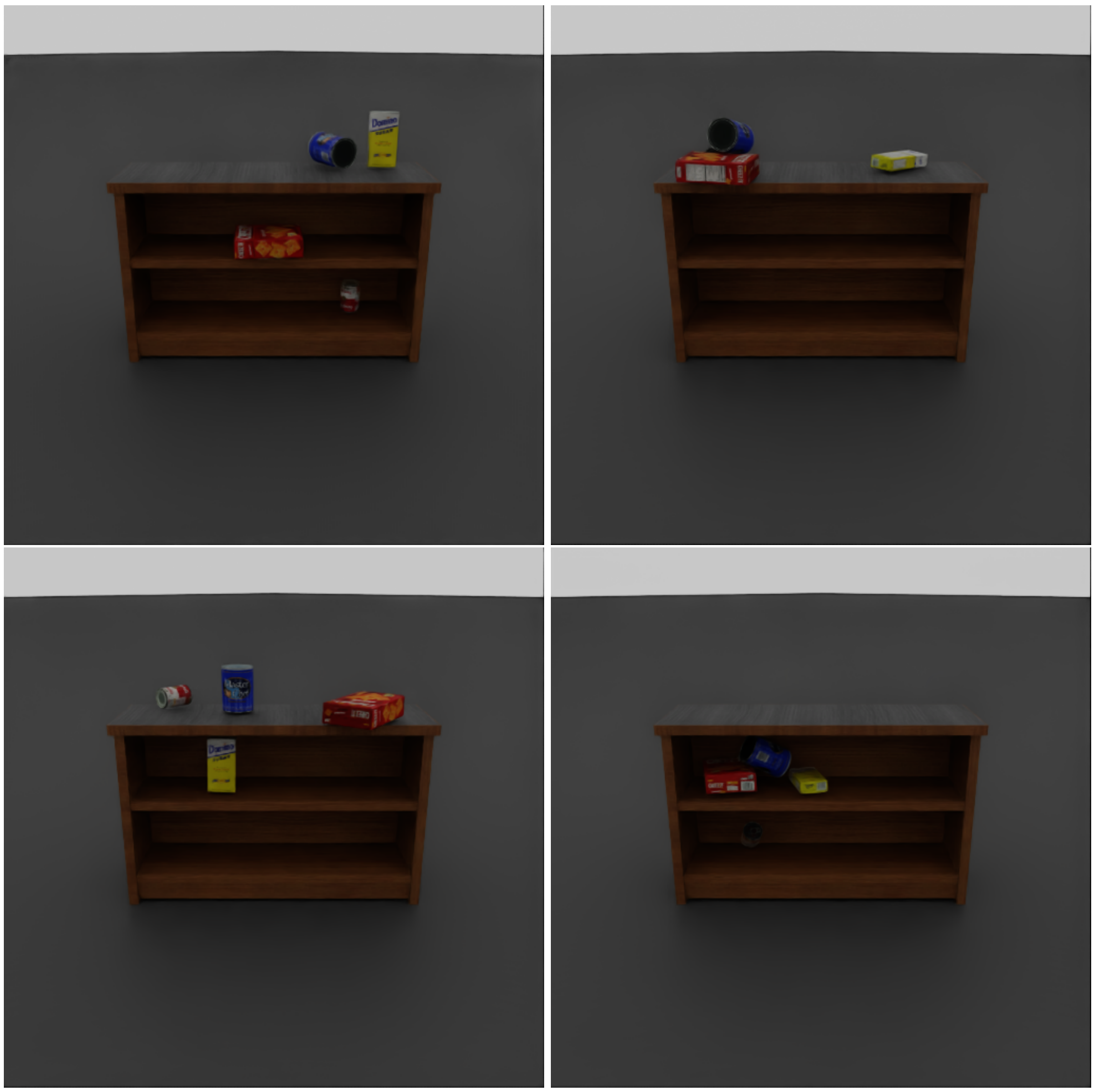

Figure 4.1: Four examples of stable and realistic pose diffusion.

We observe this functions poorly as a guidance function, appearing either to not guide the objects correctly, or to guide randomly.

We also set up a qualitative examination of the embedding space. We select several images and poses from our dataset, and write a suite of language descriptions with varying degrees of accuracy. These sentences vary, some including semantic information from the scene, some including correct relational information, and some incorrect relational information. Then we embed the image and descriptions using CLIP, and we embed the poses using our learned model.

Following this procedure, it becomes apparent that CLIP doesn't well distinguish our test cases. Sentences such as "Mustard to the left of the can" and "Mustard to the right of the can" score almost identically regardless of the ground truth of the image. This is similar to the results of [50], where they found CLIP scores for purely semantic similarity and disregarded sentence structure or relational information. They also introduced a model that sought to account for these flaws, called NegCLIP.

Building on that, further works have been published, such as Structure-CLIP [51], the benchmark Winoground [52], and SeeTRUE [53]. Of these though the only paper with available code and weights was NegCLIP. Again this had similar issues to CLIP, where it experimentally did not work and exhibited similar failures to differentiate in the embedding space.

### 4.2.2 - LANGUAGE TO LANGUAGE MODELS

For language to language models, we set up a similar comparison using Sentence-BERT [43]. We wrote ground truth language descriptions for our dataset, and wrote a suite of test language descriptions with varying degrees of accuracy. We then embedded the descriptions. This embedding space exhibits similar issues to the embeddings created by CLIP, where it is unable to distinguish the nuances required for the method to work, where leftness and rightness are much more important than the words used to describe each object. This again is presumably because of the training objective, where semantic similarity matters in largely every training example and not the physical similarity of a scene.

Alternate sentence similarity modeling methods do provide significant improvements over this, but limit our ability to train a function to match one side of the comparison. Cross encoding methods, such as [54], that compare two sentences directly can differentiate sentences with relational information much better, but don't have a shared embedding space we can map to. Since generating text at test or application time is much more difficult, this makes this approach less practical.

### 4.2.3 - RELATIONAL GUIDANCE

For testing relational guidance, following our method as described in Section 3.3.3, we sample 16 examples from our pose diffusion model with hand tuned Universal Guidance hyperparameters. We experiment with moving all objects above one, or to the left side of our environmental object using these hard-coded relational guidance functions.

Following this, we conclude relational guidance can work well. When hard-coding guidance functions, we can achieve clear success in terms of the guidance applying to the scene. We can obtain the benefits of diffusion trained for stability, our generations do seem to be stable and non-interpenetrating. We also satisfy the constraints specified by the guidance functions, where the objects satisfy all specified relations.

Some implementation details prevent larger scale experiments, where large language models struggle to pick out exactly the appropriate relations for the scene, and open set vocabulary detection on a single view struggles to identify objects, especially in orientations that don't reveal much of the objects detail. A can on its side with the bottom facing the camera for example, doesn't reveal if it contains food, pet food, paint, etc. Thus, it can be difficult to classify or order.

# 5 - DISCUSSION

## 5.1 - POSE DIFFUSION

Diffusion as a method of generative modeling seems to capture well the aspects of state we are after. Broadening this idea beyond pose, we could capture more aspects of state that are relevant to goals. Whether or not a piece of clothing is folded, if a carrot is supposed to become chopped carrot, if a cup should be filled with water, etc., are all attributes of state not easily captured by pose. Diffusion may still be a good choice when capturing non-rigid aspects of state because of its ability to model multi-modal distribution, and maintain these modes in generation.

One limitation currently is data. Capturing these aspects of state may be possible and beneficial, but it may be difficult to gather the necessary data. Furthermore, how exactly those aspects of state should be captured and once generated, used, is also not determined.

## 5.2 - GUIDANCE

Beyond generating new states, we are primarily interested in guided generations. Especially guided by language. The problem we face now, with models not largely trained on spatial information, may continue to be an issue.

### 5.2.1 - VISION LANGUAGE MODELS

There are promising results with new vision language models. Yuksekgonul et al. [50] has brought in a newer area of research, and the papers that build on this idea, such as Structure-CLIP [51], the benchmark Winoground [52], and SeeTRUE [53] may help provide better guidance.

Additionally, it might be possible to guide based on multi-modal large language models which also encode vision. The guidance function would have to backpropagate through a much larger space and would likely become too time inefficient to use as a guidance function, but with better optimization this could be an avenue worthy of exploration.

### 5.2.2 - LANGUAGE TO LANGUAGE MODELS

There may be room for future exploration in this area, but it is limited by the need to generate text descriptions at training time. This both limits what is possible to guide, and limits how scenes can be captured. It is difficult to describe things simply, without using a photo, and especially programatically.

Though it may not be worth exploring, it may be possible to fine-tune the bi-encoding methods that work for guidance using data generated with the cross-encoders that work for the sentence to sentence comparisons that are meaningful to us.

### 5.2.3 - RELATIONAL GUIDANCE

Relational guidance, while it seems to function the best of each of the methods, does come with limitations. There are many scenes which are much more difficult to describe with relations. Some scenes may require more niche relations, such as objects arranged in a circle, or arrangements where the specific orientation of an individual object is important. Adding more relations may be possible and valuable, however ensuring all scenes are well described could pose significant challenges.

Additionally, generating the relational descriptions with a Large Language Model has proven to be non-trivial, as what relations are created from a given task aren't always simple. Additionally, we may require relations between specific environment segments, or the whole environment. These implementation complexities add to the cost of the method, and may limit its impact or usefulness.

## BIBLIOGRAPHY

[1] C. R. Garrett, R. Chitnis, R. M. Holladay, *et al.*, “Integrated Task and Motion Planning,” *CoRR*, vol. abs/2010.01083, 2020, arXiv: 2010.01083. [Online]. Available: `https://arxiv.org/abs/2010.01083`.

[2] A. Bansal, H.-M. Chu, A. Schwarzschild, *et al.*, “Universal Guidance for Diffusion Models,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2023, pp. 843–852. DOI: `10.1109/CVPRW59228.2023.00091`.

[3] W. Liu, Y. Du, T. Hermans, S. Chernova, and C. Paxton, “StructDiffusion: Language-Guided Creation of Physically-Valid Structures using Unseen Objects,” in *RSS 2023*, 2023.

[4] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 2017.

[5] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [cs], Jan. 2017. DOI: `10.48550/arXiv.1412.6980`. [Online]. Available: `http://arxiv.org/abs/1412.6980` (visited on 03/30/2024).

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014, ISSN: 1533-7928. [Online]. Available: `http://jmlr.org/papers/v15/srivastava14a.html` (visited on 03/30/2024).

[7] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," en, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ISSN: 1938-7228, JMLR Workshop and Conference Proceedings, Mar. 2010, pp. 249–256. [Online]. Available: `https://proceedings.mlr.press/v9/glorot10a.html` (visited on 03/30/2024).

[8] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer Normalization*, arXiv:1607.06450 [cs, stat], Jul. 2016. DOI: `10.48550/arXiv.1607.06450`. [Online]. Available: `http://arxiv.org/abs/1607.06450` (visited on 03/30/2024).

[9] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, arXiv:1512.03385 [cs], Dec. 2015. DOI: `10.48550/arXiv.1512.03385`. [Online]. Available: `http://arxiv.org/abs/1512.03385` (visited on 03/30/2024).

[10] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention Is All You Need," *CoRR*, vol. abs/1706.03762, 2017, arXiv: 1706.03762. [Online]. Available: `http://arxiv.org/abs/1706.03762`.

[11] T. B. Brown, B. Mann, N. Ryder, *et al.*, *Language Models are Few-Shot Learners*, arXiv:2005.14165 [cs], Jul. 2020. DOI: `10.48550/arXiv.2005.14165`. [Online]. Available: `http://arxiv.org/abs/2005.14165` (visited on 04/07/2024).

[12] Y. Jiang, A. Gupta, Z. Zhang, *et al.*, "VIMA: General Robot Manipulation with Multimodal Prompts," in *Fortieth International Conference on Machine Learning*, 2023.

[13] M. Shridhar, L. Manuelli, and D. Fox, "CLIPort: What and Where Pathways for Robotic Manipulation," in *Conference on Robot Learning, 8-11 November 2021, London, UK*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. Proceedings of Machine Learning Research, vol. 164, PMLR, 2021, pp. 894–906. [Online]. Available: `https://proceedings.mlr.press/v164/shridhar22a.html`.

[14] C. Chi, S. Feng, Y. Du, *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[15] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-Actor: A Multi-Task Transformer for Robotic Manipulation," in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[16] W. Liu, C. Paxton, T. Hermans, and D. Fox, "StructFormer: Learning Spatial Structure for Language-Guided Semantic Rearrangement of Novel Objects," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 6322–6329. DOI: `10.1109/ICRA46639.2022.9811931`.

[17] P.-L. Guhur, S. Chen, R. G. Pinel, M. Tapaswi, I. Laptev, and C. Schmid, "Instruction-driven history-aware policies for robotic manipulations," in *Proceedings of The 6th Conference on Robot Learning*, K. Liu, D. Kulic, and J. Ichnowski, Eds., ser. Proceedings of Machine Learning Research, vol. 205, PMLR, Dec. 2023, pp. 175–187. [Online]. Available: `https://proceedings.mlr.press/v205/guhur23a.html`.

[18] C. Paxton, C. Xie, T. Hermans, and D. Fox, "Predicting Stable Configurations for Semantic Placement of Novel Objects," in *Proceedings of the 5th Conference on Robot*

*Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. Proceedings of Machine Learning Research, vol. 164, PMLR, Nov. 2022, pp. 806–815. [Online]. Available: `https://proceedings.mlr.press/v164/paxton22a.html`.

[19] S. Huang, Z. Wang, P. Li, *et al.*, “Diffusion-based Generation, Optimization, and Planning in 3D Scenes,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[20] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, “Planning with Diffusion for Flexible Behavior Synthesis,” in *International Conference on Machine Learning*, 2022.

[21] I. Kapelyukh, V. Vosylius, and E. Johns, “DALL-E-Bot: Introducing Web-Scale Diffusion Models to Robotics,” *IEEE Robotics and Automation Letters (RA-L)*, 2023.

[22] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, “Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents,” *ICML*, 2022.

[23] M. Ahn, A. Brohan, N. Brown, *et al.*, “Do As I Can and Not As I Say: Grounding Language in Robotic Affordances,” in *arXiv preprint arXiv:2204.01691*, 2022.

[24] A. Zeng, M. Attarian, B. Ichter, *et al.*, *Socratic Models: Composing Zero-Shot Multimodal Reasoning with Language*, _eprint: 2204.00598, 2022.

[25] W. Huang, F. Xia, T. Xiao, *et al.*, “Inner Monologue: Embodied Reasoning through Planning with Language Models,” in *arXiv preprint arXiv:2207.05608*, 2022.

[26] J. Liang, W. Huang, F. Xia, *et al.*, “Code as Policies: Language Model Programs for Embodied Control,” in *arXiv preprint arXiv:2209.07753*, 2022.

[27] I. Singh, V. Blukis, A. Mousavian, *et al.*, “ProgPrompt: Generating Situated Robot Task Plans using Large Language Models,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 523–11 530. DOI: `10.1109/ICRA48891.2023.10161317`.

[28] M. Ghallab, C. Knoblock, D. Wilkins, *et al.*, “PDDL - The Planning Domain Definition Language,” Aug. 1998.

[29] B. Liu, Y. Jiang, X. Zhang, *et al.*, *LLM+P: Empowering Large Language Models with Optimal Planning Proficiency*, arXiv:2304.11477 [cs], Sep. 2023. DOI: `10.48550/arXiv.2304.11477`. [Online]. Available: `http://arxiv.org/abs/2304.11477` (visited on 04/07/2024).

[30] Y. Chen, J. Arkin, C. Dawson, Y. Zhang, N. Roy, and C. Fan, *AutoTAMP: Autoregressive Task and Motion Planning with LLMs as Translators and Checkers*, arXiv:2306.06531 [cs], Mar. 2024. DOI: `10.48550/arXiv.2306.06531`. [Online]. Available: `http://arxiv.org/abs/2306.06531` (visited on 04/07/2024).

[31] Y. Liu, L. Palmieri, S. Koch, I. Georgievski, and M. Aiello, *DELTA: Decomposed Efficient Long-Term Robot Task Planning using Large Language Models*, arXiv:2404.03275 [cs], Apr. 2024. DOI: `10.48550/arXiv.2404.03275`. [Online]. Available: `http://arxiv.org/abs/2404.03275` (visited on 04/07/2024).

[32] J. Ho, A. Jain, and P. Abbeel, “Denoising Diffusion Probabilistic Models,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020,

pp. 6840–6851. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[33] J. Song, C. Meng, and S. Ermon, “Denoising Diffusion Implicit Models,” *CoRR*, vol. abs/2010.02502, 2020, arXiv: 2010.02502. [Online]. Available: https://arxiv.org/abs/2010.02502.

[34] A. Simeonov, A. Goyal, L. Manuelli, *et al.*, “Shelving, Stacking, Hanging: Relational Pose Diffusion for Multi-modal Rearrangement,” *Conference on Robot Learning*, 2023.

[35] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation,” *CoRR*, vol. abs/1612.00593, 2016, arXiv: 1612.00593. [Online]. Available: http://arxiv.org/abs/1612.00593.

[36] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space,” *CoRR*, vol. abs/1706.02413, 2017, arXiv: 1706.02413. [Online]. Available: http://arxiv.org/abs/1706.02413.

[37] W. Wu, Z. Qi, and F. Li, “PointConv: Deep Convolutional Networks on 3D Point Clouds,” *CoRR*, vol. abs/1811.07246, 2018, arXiv: 1811.07246. [Online]. Available: http://arxiv.org/abs/1811.07246.

[38] H. Zhao, L. Jiang, J. Jia, P. H. S. Torr, and V. Koltun, “Point Transformer,” *CoRR*, vol. abs/2012.09164, 2020, arXiv: 2012.09164. [Online]. Available: https://arxiv.org/abs/2012.09164.

[39] X. Wu, Y. Lao, L. Jiang, X. Liu, and H. Zhao, *Point Transformer V2: Grouped Vector Attention and Partition-based Pooling*, _eprint: 2210.05666, 2022.

[40] W. Wu, L. Fuxin, and Q. Shan, "PointConvFormer: Revenge of the Point-based Convolution," in *CVPR*, 2023. [Online]. Available: `https://arxiv.org/abs/2208.02879`.

[41] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the Continuity of Rotation Representations in Neural Networks," *CoRR*, vol. abs/1812.07035, 2018, arXiv: 1812.07035. [Online]. Available: `http://arxiv.org/abs/1812.07035`.

[42] A. Radford, J. W. Kim, C. Hallacy, *et al.*, "Learning Transferable Visual Models From Natural Language Supervision," *CoRR*, vol. abs/2103.00020, 2021, arXiv: 2103.00020. [Online]. Available: `https://arxiv.org/abs/2103.00020`.

[43] N. Reimers and I. Gurevych, *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*, arXiv:1908.10084 [cs], Aug. 2019. DOI: `10.48550/arXiv.1908.10084`. [Online]. Available: `http://arxiv.org/abs/1908.10084` (visited on 04/11/2024).

[44] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, arXiv:1810.04805 [cs], May 2019. DOI: `10.48550/arXiv.1810.04805`. [Online]. Available: `http://arxiv.org/abs/1810.04805` (visited on 04/11/2024).

[45] Y. Huang, N. C. Taylor, A. Conkey, W. Liu, and T. Hermans, *Latent Space Planning for Multi-Object Manipulation with Environment-Aware Relational Classifiers*, _eprint: 2305.10857, 2023.

[46] Y. Huang, A. Conkey, and T. Hermans, *Planning for Multi-Object Manipulation with Graph Neural Network Relational Classifiers*, _eprint: 2209.11943, 2023.

[47] S. Minaee, T. Mikolov, N. Nikzad, *et al.*, *Large Language Models: A Survey*, arXiv:2402.06196 [cs], Feb. 2024. DOI: `10.48550/arXiv.2402.06196`. [Online]. Available: `http://arxiv.org/abs/2402.06196` (visited on 04/12/2024).

[48] M. Deitke, D. Schwenk, J. Salvador, *et al.*, "Objaverse: A universe of annotated 3d objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 142–13 153.

[49] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The YCB object and model set: Towards common benchmarks for manipulation research," in *ICAR*, 2015.

[50] M. Yuksekgonul, F. Bianchi, P. Kalluri, D. Jurafsky, and J. Zou, "When and why Vision-Language Models behave like Bags-of-Words, and what to do about it?" In *International Conference on Learning Representations*, 2023. [Online]. Available: `https://openreview.net/forum?id=KRLUvxh8uaX`.

[51] Y. Huang, J. Tang, Z. Chen, *et al.*, *Structure-CLIP: Towards Scene Graph Knowledge to Enhance Multi-modal Structured Representations*, _eprint: 2305.06152, 2023.

[52] T. Thrush, R. Jiang, M. Bartolo, *et al.*, "Winoground: Probing Vision and Language Models for Visio-Linguistic Compositionality," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 5228–5238. DOI: `10.1109/CVPR52688.2022.00517`.

[53] M. Yarom, Y. Bitton, S. Changpinyo, *et al.*, *What You See is What You Read? Improving Text-Image Alignment Evaluation*, arXiv:2305.10400 [cs], Dec. 2023. DOI: 10.48550/arXiv.2305.10400. [Online]. Available: http://arxiv.org/abs/2305.10400 (visited on 04/11/2024).

[54] Y. Yang, S. Qi, C. Liu, Q. Wang, C. Gao, and Z. Xu, *Once is Enough: A Light-Weight Cross-Attention for Fast Sentence Pair Modeling*, arXiv:2210.05261 [cs], Oct. 2023. DOI: 10.48550/arXiv.2210.05261. [Online]. Available: http://arxiv.org/abs/2210.05261 (visited on 04/11/2024).

Name of Candidate: Andrew Nichols Crawford Taylor

Date of Submission: April 26, 2024