Robotics: Science and Systems 2022
New York City, NY, USA, June 27-July 1, 2022

Data Augmentation for Manipulation

Peter Mitrano
University of Michigan
Email: pmitrano@umich.edu

Dmitry Berenson
University of Michigan
Email: dmitryb@umich.edu

***Abstract*—The success of deep learning depends heavily on the availability of large datasets, but in robotic manipulation there are many learning problems for which such datasets do not exist. Collecting these datasets is time-consuming and expensive, and therefore learning from small datasets is an important open problem. Within computer vision, a common approach to a lack of data is *data augmentation*. Data augmentation is the process of creating additional training examples by modifying existing ones. However, because the types of tasks and data differ, the methods used in computer vision cannot be easily adapted to manipulation. Therefore, we propose a data augmentation method for robotic manipulation. We argue that augmentations should be valid, relevant, and diverse. We use these principles to formalize augmentation as an optimization problem, with the objective function derived from physics and knowledge of the manipulation domain. This method applies rigid body transformations to trajectories of geometric state and action data. We test our method in two scenarios: 1) learning the dynamics of planar pushing of rigid cylinders, and 2) learning a constraint checker for rope manipulation. These two scenarios have different data and label types, yet in both scenarios, training on our augmented data significantly improves performance on downstream tasks. We also show how our augmentation method can be used on real-robot data to enable more data-efficient online learning.**

I. Introduction

In recent years, interest in applying deep learning to robotic manipulation has increased. However, the lack of cheap data has proven to be a significant limitation [23]. To enable applications such as smart and flexible manufacturing, logistics, and care-giving robots [21], we must develop methods that learn from smaller datasets, especially if the learning is done online on real robots.

One of the simplest and most effective ways to mitigate the problem of small datasets is to use data augmentation. While data augmentation has been shown to significantly improve generalization performance in tasks like image classification, it is not straightforward to extend existing data augmentation methods to the types of data used in robotic manipulation. Furthermore, most existing augmentation methods fall into one of two categories, and both have severe limitations:

In the first category, augmentations are defined by a set of transformations, sampled independently for each example. Most image augmentation methods fall into this category, where rotations or crops are sampled randomly for each example [1, 5, 17]. By making augmentations independent of the example being augmented, we are restricted to operations which are valid on all examples. In the second category, there are methods which learn a generative model (VAE, GAN, etc.) of the data, and then sample new training examples from that model [19, 15, 2]. This approach assumes a useful generative model can be learned from the given dataset, but we found these methods did not perform well when the dataset is small.

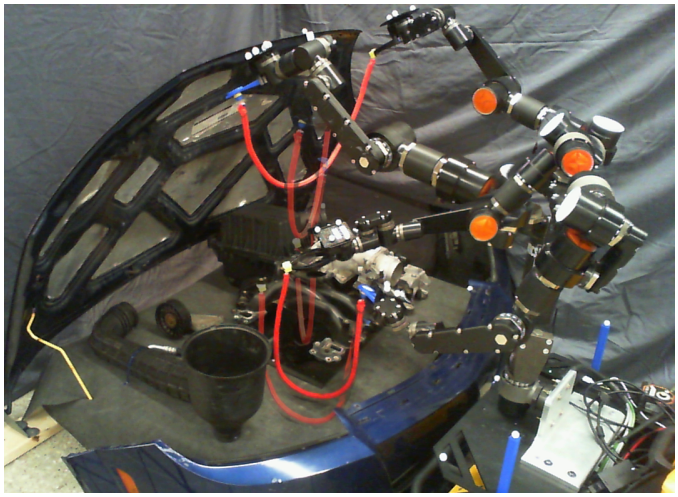

Fig. 1: A mock-up of a car engine bay. The robot must move the rope and place it under the engine without snagging it to set up for lifting the engine. We use data augmentation to improve task success rate during online learning for this task.

It is not trivial to define a coherent framework for data augmentation that encompasses many domains and many types of learning problems (e.g. classification and regression). Thus, the first contribution of this paper is a formalization of the data augmentation problem. In our problem statement (Section III), we formalize data augmentation as an optimization problem based on three key criteria: *validity*, *relevance*, and *diversity*. We define an augmented example as *valid* if it obeys the laws of physics and is possible in the real world. Augmentations are *relevant* if they are similar to data that would be seen when performing the target task. *Diversity* encourages the augmentations to be as varied as possible, i.e. the transformations applied to the data should be uniformly distributed to maximize diversity. Producing diverse augmentations for each original example is key to improving the generalization of the trained network.

The general definitions of validity, relevance, and diversity we propose depend on information that is intractable to compute for many manipulation problems, and therefore we also present approximations to these definitions. We do not claim that this formulation is useful for all manipulation problems, and clearly define the physical assumptions behind

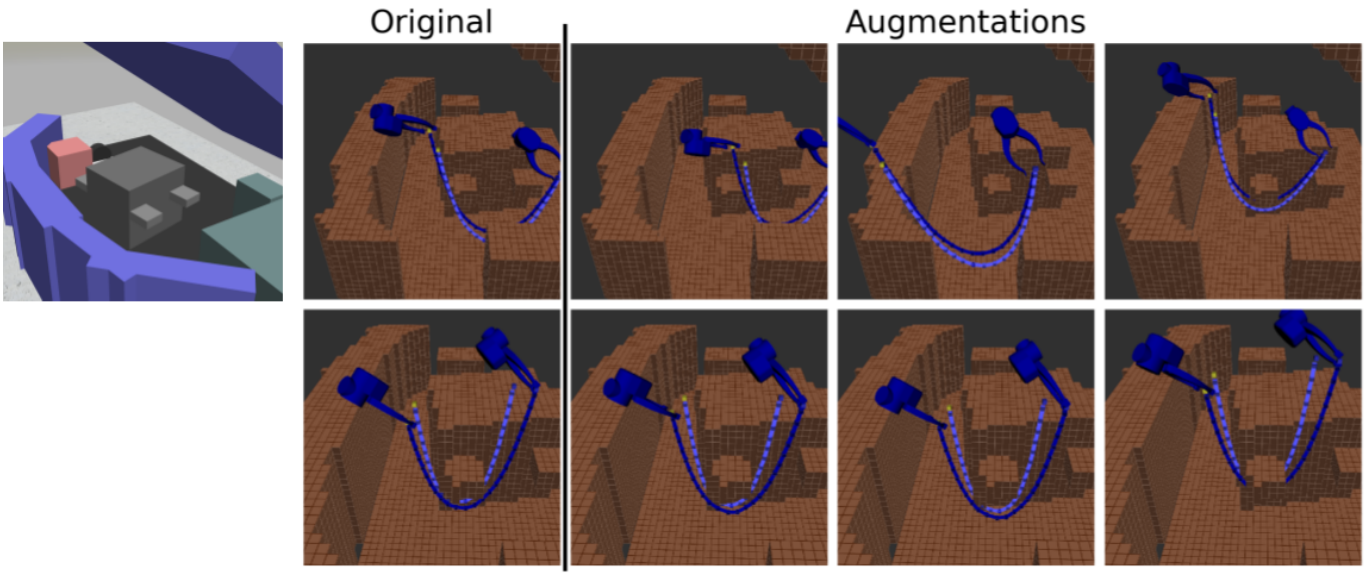

Fig. 2: Examples of augmentations of rope generated by our method. On the left is a picture of the scene in simulation from a zoomed out viewpoint. The simplified engine block model is in the center. The rope start (dark blue) and end (light blue) states are shown, with the grippers shown at the start state. The static environment geometry is shown in brown. The first row shows a transition in free space, where the resulting augmentations are particularly diverse. The final augmentation shows how our method found a transformation to move the rope underneath the hook while remaining in free space. The second row shows a transition which involves contact between the rope and the environment. The augmentations preserve this contact.

this formulation in Section III.

Our second contribution is a method for solving this approximated optimization problem. Our method operates on trajectories of object poses and velocities, and searches for rigid-body transformations to apply to the moving objects in the scene to produce augmentations. Our method encourages validity by preserving contacts and the influence of gravity. Additionally, we encourage relevance by initializing the augmentations nearby the original examples and preserving near-contacts. Finally, we encourage diversity by pushing the augmentations towards randomly sampled targets.

Our results demonstrate that training on our augmentations improves downstream task performance for a simulated cluttered planar-pushing task and a simulated bimanual rope manipulation task. The learning problems in these tasks include classification and regression, and have high-dimensional inputs and outputs. Lastly, we demonstrate our augmentation in an online learning scenario on real-robot bimanual rope manipulation using noisy real-world perception data (Figure 1). In this scenario, augmentation increased the success rate from 27% to 50% in only 30 trials. Additional materials such as code and video can be found on our project website.

II. Related Work

This paper studies the problem of learning better models from limited data, which is important for robotics applications and has received significant attention as researchers have tried to apply deep learning to robotics [23, 8].

Data augmentation has been applied to many machine learning problems, from material science [15] to financial modeling [2] (see [6, 3, 16] for several surveys). It is especially common in computer vision [16, 17, 5, 9, 1], and is also popular in natural language processing [3, 11]. In these fields the data is often in standardized data types—images, text, or vectors of non-physical features (e.g. prices). Each of the data types can be used for a wide variety of tasks, and various data augmentations have been developed for various pairings of data type and task.

However, problems in robotic manipulation use other formats of data, such as point clouds, or object poses, and may consist of time-series data mixed with time-invariant data. To the best of our knowledge, there are currently no augmentation methods designed specifically for data of the types above. Our proposed method is intended to fill this gap.

In contrast to engineering augmentations based on prior knowledge, another body of work uses unsupervised generative models to generate augmentations [19, 15, 2]. Typically, these methods train a model like an Auto-Encoder or Generative Adversarial Network (GAN) [4]) on the data, encode the input data into the latent space, perturb the data in the latent space, then decode to produce the augmented examples. These methods can be applied to any data type, and handle both regression and classification problems. However, they do not incorporate prior knowledge, and only add small but sophisticated noise. In contrast, we embed prior knowledge about the physical and spatial nature of manipulation, and as a result can produce large and meaningful augmentations, at the cost of being less generally-applicable.

Although data augmentation is a popular approach, there are many other methods that have been proposed for data-

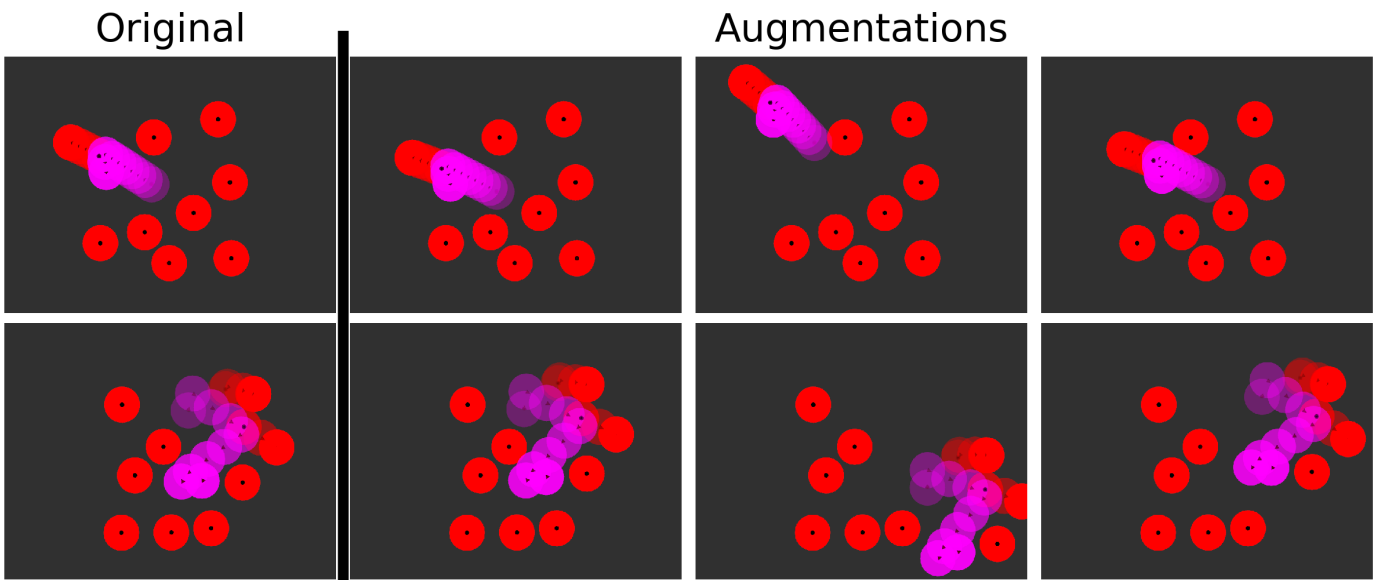

Fig. 3: Examples of augmentations generated for learning the dynamics of planar pushing of 9 cylinders. The pink cylinder is the robot. Time is indicated by transparency. Augmentation transforms the positions and velocities of the cylinders that moved, including the robot. All moved objects are transformed together, rigidly. Despite the clutter, we are able to find relatively large transformations that still preserve existing contacts but do not create any new ones.

efficient learning, both in general and specifically for robotic manipulation. We highlight a few important ones here, but note that these are all complementary to data augmentation. The most common technique is simply to pick a low-capacity model class, such as linear models or very small neural networks [14, 24]. Alternatively, prior work has also developed various sets of priors specific to robotics [7] which can be used as objectives during training. Another extremely useful technique is to engineer the state or action representations to include certain known invariances. For instance, a standard technique in dynamics learning methods is to represent the input positions in a local frame as opposed to a world frame, to encode translation invariance [10]. There are also methods for learning these kinds of invariances [24].

Finally, there are methods which automatically tune the parameters of a space of manually-defined augmentations, such as rotations and crops [1, 5]. These techniques are also compatible with our proposed method, and could be used to tune the hyperparameters of our algorithm.

III. Problem Statement

In this section, we formally define the form of data augmentation studied in this paper. We define a dataset $\mathcal{D}$ as a list of examples $x \in \mathcal{X}$ and, optionally, labels $\ell(x) = w$, where ℓ is a task-specific labeling function. We assume the space $\mathcal{X}$ is a metric space with a distance function dist. Augmentation is a stochastic function $\phi : \mathcal{X} \to \mathcal{X}$ which takes in an example x and produces the augmented example $\tilde{x}$. The general form is shown in Algorithm 1. Internally, augmentation will call `sample` to generate a vector of parameters, which we call T. We also define $\tilde{x}_{1:k}$ as a set of k augmented examples produced by calling `sample` then `apply` k times.

Algorithm 1: $\phi(x)$

1 $T = \texttt{sample}(x)$
2 $\tilde{x} = \texttt{apply}(x, T)$
3 return $\tilde{x}$

The parameters T describe the transformation which will be applied to the example in the `apply` procedure. We focus on augmentation functions that are stochastic, thus ϕ will sample new augmented examples each time it is called. If the dataset contains labels w, we assume that the labels should not change when the example is augmented.

We propose that useful augmentations should be *valid*, *relevant*, and *diverse*. Let the valid set $\mathcal{X}_v$ be the set of examples which are physically possible. Let the relevant set $\mathcal{X}_r$ be the set of examples likely to occur when collecting data for or executing a specific set of tasks in a specific domain. We define $\text{validity}(\tilde{x}) = 1$ if $\tilde{x} \in \mathcal{X}_v$ and $\text{validity}(\tilde{x}) = 0$ otherwise. We also define $\text{relevance}(\tilde{x}) := e^{-\text{dist}(\tilde{x}, \mathcal{X}_r)}$ and $\text{diversity}(\tilde{x}_{1:k}) := e^{-D_{KL}(p_{\tilde{x}_{1:k}}(T) \,||\, \mathbb{U}[T^-, T^+])}$, where D_{KL} is the Kullback–Leibler divergence and $p_{\tilde{x}_{1:k}}(T)$ is the distribution of the parameters for a set of augmented examples $\tilde{x}_{1:k}$. Diversity is maximized when the augmentation transformations are uniformly distributed in the range $[T^-, T^+]$. With these concepts defined, we define data augmentation as the following optimization problem, the solution to which is a set of augmentations $\tilde{x}_{1:k}$:

$$\begin{array}{ll} \max\limits_{\tilde{x}_{1:k}} & \text{diversity}(\tilde{x}_{1:k}) + \beta \sum_{\tilde{x}_i} \text{relevance}(\tilde{x}_i) \\ \text{subject to} & \text{validity}(\tilde{x}_i) \quad \forall\, \tilde{x}_i \in \tilde{x}_{1:k} \\ & \ell(\tilde{x}) = \ell(x) \end{array} \tag{1}$$

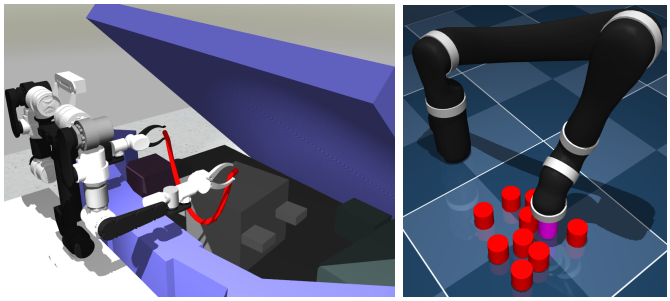

Fig. 4: (left) The environment for bimanual rope manipulation, in simulation. (right) The environment for cluttered planar pushing of cylinders, in simulation.

where β is a positive scalar.

This optimization problem can be solved directly if $\mathcal{X}_v$, $\mathcal{X}_r$, and ℓ are known. However, in manipulation tasks, that is rarely the case. Instead, we will formulate an approximation to this problem using measures of relevance, diversity, and validity that are derived from physics and useful for a variety of robotic manipulation tasks and domains.

A. Assumptions

Most augmentation algorithms rely on some expert knowledge or heuristics to define what is a valid augmentation. For instance, rotating an image for image classification makes an assumption that rotation does not change the label, and this is not always true. Similarly, the efficacy or correctness of our algorithm is also subject to certain assumptions. Here, we define the key assumptions:

- The geometry of the robot and all objects is known.
- The scene can be decomposed into objects which can be assigned or detected as either moving or stationary.
- Examples are time-series, consisting of at least two states.
- All possible contacts between stationary vs. moving objects have the same friction coefficient.
- Contacts between the robot and objects/environment (e.g. grasps) can be determined from the data.
- A rigid-body transformation of an object preserves internal forces arising from its material properties.
- Objects only move due to contact or under the force of gravity. We do not handle movement due to magnetism or wind, for example.

Notably, the assumption that a rigid-body transformation preserves internal, material forces is what allows us to handle cluttered scenes with many moving objects, as well as deformable or articulated objects. While it could be valuable to augment the deformation or relative motion of the objects, doing so in a way that is valid would be challenging. Instead, we transform them all rigidly (See examples in Figures 2,3).

The assumption of having a common friction coefficient between all moving versus stationary objects is in-line with much manipulation research. For example, work on planar pushing assumes friction is uniform across the plane [25, 22]. Note that we make no assumption on the coefficients of friction between two moving objects.

Naturally, there are scenarios where these assumptions do not hold and thus where our algorithm may not perform well. However, our experiments demonstrate significant improvement on two very different manipulation scenarios, and we expect these assumptions extend to other scenarios as well.

IV. Methods

We first describe an approximation to the augmentation problem (1), which is specialized for manipulation. Next, we decompose this problem and describe each component in detail.

A. Algorithm Overview

Since robotic manipulation is interested specifically in moving objects, we focus on augmenting trajectories of poses and velocities of moving objects. A key insight is that objects in the scene can be categorized as either robots, moved objects, or stationary objects, and that these should be considered differently in augmentation. We denote the moved objects state as s, the robot state as r, the robot action as a, and the stationary objects as e (also called environment). Our method augments the moved object states, the robot state, and the actions, but not the stationary objects. We do not assume any specific representation for states or actions, and examples of possible representations include sets of points, joint angles, poses, or joint velocities. Since we operate on trajectories, we bold the state $(\boldsymbol{s}, \boldsymbol{r})$ and action $(\boldsymbol{a})$ variables to indicate time-series (e.g $s_{1:T} = \boldsymbol{s}$). With this categorization, we can write $x = \{\boldsymbol{s}, \boldsymbol{r}, \boldsymbol{a}, e\}$ and $\tilde{x} = \{\tilde{\boldsymbol{s}}, \tilde{\boldsymbol{r}}, \tilde{\boldsymbol{a}}, e\}$.

We choose the parameters T to be rigid body transformations, i.e. either $SE(2)$ or $SE(3)$. We parameterize T as a vector with translation and rotation components, with the rotation component with Euler angles bounded from $-\pi/2$ to $\pi/2$, which gives uniqueness and a valid distance metric. These rigid body transforms are applied to moved objects in the scene, and augmented robot state and action are computed to match. We choose rigid body transforms because we can reasonably assume that even for articulated or deformable objects, augmenting with rigid body transforms preserve the internal forces, and therefore the augmentations are likely to be valid.

It may seem that an effective method to generate augmentations is then to randomly sample transforms independent of the data. However, this is not an effective strategy because it is highly unlikely to randomly sample valid and relevant transformations. We confirm this in our ablations studies (included in the Appendix 1.A). Instead of sampling transforms randomly, we formulate an approximation to Problem 1:

$$\begin{aligned} \min_{T} \quad & \mathcal{L}_{\mathbb{U}}(T, T^{\text{target}}) + \beta_1 \mathcal{L}_{\text{bbox}}(\tilde{\boldsymbol{s}}) + \beta_2 \mathcal{L}_{\text{valid}}(T) + \\ & \beta_3 \mathcal{L}_{\text{occ}}(\tilde{\boldsymbol{s}}, e) + \beta_4 \mathcal{L}_{\Delta d^-}(\tilde{\boldsymbol{s}}, e) + \\ & \mathcal{L}_{\text{robot}}(\tilde{\boldsymbol{s}}, \tilde{\boldsymbol{r}}, \tilde{\boldsymbol{a}}, e) \\ \text{subject to} \quad & \{\tilde{\boldsymbol{s}}, \tilde{\boldsymbol{r}}, \tilde{\boldsymbol{a}}, e\} = \texttt{apply}(\boldsymbol{s}, \boldsymbol{r}, \boldsymbol{a}, e, T) \\ & T^{\text{target}} \sim \mathbb{U}[T^-, T^+] \end{aligned} \tag{2}$$

The decision variable is now the parameters T, and the validity constraint is moved into the objective. We propose that diversity should be maximized by the transforms being uniformly distributed, and therefore $\mathcal{L}_{\mathbb{U}}$ penalizes the distance to a target transform T^{target} sampled uniformly within $[T^-, T^+]$. The relevance and validity terms (which are intractable to compute) are replaced with four objective functions, which are specialized to manipulation. The magnitudes of different terms are balanced by $\beta_1, \beta_2, \beta_3, \beta_4$, which are defined manually. We define each objective function below:

1) Bounding Box Objective: First is the bounding-box objective $\mathcal{L}_{\text{bbox}}$, which keeps the augmented states $\tilde{\boldsymbol{s}}$ within the workspace/scene bounds defined by $[s^-, s^+]$. The bounding box objective encourages relevance, since states outside the workspace are unlikely to be relevant for the task.

$$\mathcal{L}_{\text{bbox}} = \sum_{i=1}^{|s|} \max(0, \tilde{\boldsymbol{s}}_i - s_i^+) + \max(0, s_i^- - \tilde{\boldsymbol{s}}_i) \tag{3}$$

2) Transformation Validity Objective: The transformation validity objective $\mathcal{L}_{\text{valid}}$ assigns high cost to transformations that are always invalid or irrelevant for the particular task or domain. It is defined by function f_{valid}, which takes in only the transformation. For example, in our rope manipulation case, it is nearly always invalid to rotate the rope so that it floats sideways. In our cluttered pushing task, in contrast, this term has no effect. This term can be chosen manually on a per-task basis, but we also describe how a transformation validity objective can be learned from data in section IV-C.

$$\mathcal{L}_{\text{valid}} = f_{\text{valid}}(T) \tag{4}$$

3) Occupancy Objective: The occupancy objective $\mathcal{L}_{\text{occ}}$ is designed to ensure validity by preventing objects that were separate in the original example from penetrating each other and ensuring that any existing penetrations are preserved. In other words, we ensure that the occupancy $O(p)$ of each point $\tilde{p}_{s,i} \in \tilde{p}_s$ in the augmented object state matches the occupancy of the corresponding original point $p_{s,i} \in p_s$. For this term, we directly define the gradient, which moves $\tilde{p}_{s,i}$ in the correct direction when the occupancies do not match. This involves converting the environment e into a signed-distance field (SDF) and the moved objects states $\boldsymbol{s}$ into points p_s. This objective assumes that the environment has uniform friction, so that a contact/penetration in one region of the environment can be moved to another region.

$$\mathcal{L}_{\text{occ}} = \sum_{\substack{p_{s,i} \in p_s \\ \tilde{p}_{s,i} \in \tilde{p}_s}} \text{SDF}(\tilde{p}_{s,i}) \big(O(p_{s,i}) - O(\tilde{p}_{s,i})\big) \tag{5}$$

4) Delta Minimum Distance Objective: The delta minimum distance objective is designed to increase relevance by preserving near-contact events in the data. We preserve near-contact events because they may signify important parts of the task, such as being near a goal object or avoiding an obstacle. We define the point among the moved object points p_s which has the minimum distance to the environment $p_{d^-} = \text{argmin}_{p_{s,i}} \text{SDF}(p_{s,i})$. The corresponding point in the augmented example we call $\tilde{p}_{d^-}$.

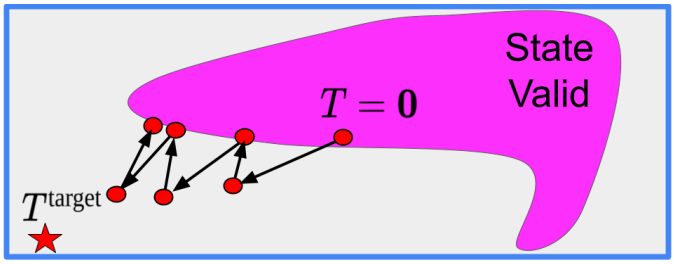

Fig. 5: Illustration of `aug_state` within Algorithm 2. All points and sets are in the space of T. The path of T is shown in red with black arrows. The pink set, State Valid, is the set where *state_valid* is true. T begins at the origin, and alternates between moving towards T^{target} and projecting back into the set *state_valid* (by solving Equation (8)).

$$\mathcal{L}_{\Delta d^-} = ||\text{SDF}(p_{d^-}) - \text{SDF}(\tilde{p}_{d^-})||_2^2 \tag{6}$$

5) Robot Contact Objective: The robot contact objective $\mathcal{L}_{\text{robot}}$ ensures validity of the robot's state and the action. This means that contacts involving the moved objects which existed in the original example must also exist in the augmented example. Let the contact points on the robot be p_r^c and the contact points on the moved objects' state be p_s^c.

$$\mathcal{L}_{\text{robot}} = \sum_i (||p_{r,i}^c - p_{s,i}^c||_2^2) \tag{7}$$

Finally, we note that other objective functions can be added for the purpose of preserving task-specific labels, i.e. so that $\ell(x) = \ell(\tilde{x})$. However, for our experiments, no additional functions were necessary.

B. Solving the Augmentation Optimization Problem

This section describes how we solve Problem (2), and the procedure is detailed in Algorithm 2. First, we split the problem into two parts, `aug_state` and `aug_robot`.

In `aug_state`, we optimize the transform T to produce the moved objects' state $\tilde{\boldsymbol{s}}$ while considering environment e. To achieve diversity, we uniformly sample a target transform T^{target} and step towards it iteratively. This stepping alternates with optimizing for validity and relevance. We visualize this procedure in Figure 5, as well as in the supplementary video. The innermost optimization problem is

$$\underset{T}{\text{argmax}} \quad \beta_1 \mathcal{L}_{\text{bbox}} + \beta_2 \mathcal{L}_{\text{valid}} + \beta_3 \mathcal{L}_{\text{occ}} + \beta_4 \mathcal{L}_{\Delta d^-} \tag{8}$$

We solve Problem (8) using gradient descent, terminating after either M_p steps or until the gradient is smaller than some threshold ϵ_p.

Note that we start `aug_state` in Algorithm 2 with T at the identity transformation, rather than initially sampling uniformly. This has two benefits. First, the identity transform gives the original example, which is always in the relevant set. Second, it is unlikely that a uniformly sampled transformation

Algorithm 2: $\phi(\boldsymbol{s}, \boldsymbol{r}, \boldsymbol{a}, e)$

```
   // aug_state
1  T^target ~ U[T^-, T^+]
2  T = 0  (identity)
3  for i ∈ N_p do
4      T_old = T
5      T = step_towards(T, T^target)
6      T = solve Equation (8)
7      if dist(T, T_old) < δ_p then
8          break
   // aug_robot
9  s̃, state_valid = apply_state(s, a, T)
10 r̃, ã, ik_valid ← IK(r, a, s̃, e)
11 if !state_valid or !ik_valid then
12     return s, r, a, e
13 else
14     return s̃, r̃, ã, e
```

Algorithm 3: Data Collection for Learning Valid Transformations

Input: $Q_{\text{valid}}, n_{\text{valid}}$
Output: $\mathcal{D}_{\text{valid}}$

1 $y^-_{\text{valid}} = \infty$
2 **for** $i \in [1, n_{valid}]$ **do**
3 **for** $(s_t, r_t, a_t, e) \in Q_{valid}$ **do**
4 $\alpha_{\text{valid}} = i / n_{\text{valid}}$
5 $T \sim \mathbb{U}[\alpha_{\text{valid}} T^-, \alpha_{\text{valid}} T^+]$
6 $s_{t+1}, r_{t+1} = \texttt{simulate}(s_t, r_t, a_t, e)$
7 $\tilde{s}_{t,t+1}, \tilde{r}_{t,t+1}, \tilde{a}_t = \texttt{apply}(s_{t,t+1}, r_{t,t+1}, a_t, T)$
8 $\tilde{s}'_{t+1}, \tilde{r}'_{t+1} = \texttt{simulate}(\tilde{s}_t, \tilde{r}_t, \tilde{a}_t, e)$
9 $y_{\text{valid}} = ||\tilde{s}_{t+1} - \tilde{s}'_{t+1}||$
10 **if** $y_{valid} < y^-_{valid}$ **then**
11 $y^-_{\text{valid}} = y_{\text{valid}}$, $T_{\text{min}} = T$, $y^-_{\text{valid}} = y_{\text{valid}}$
12 add $(T_{\text{min}}, y^-_{\text{valid}})$ to $\mathcal{D}_{\text{valid}}$
13 return $\mathcal{D}_{\text{valid}}$

is valid or relevant, so starting at a random transformation would make solving Problem (8) more difficult.

In `aug_robot`, we are optimizing $\mathcal{L}_{\text{robot}}$. This corresponds to computing the augmented robot states $\tilde{\boldsymbol{r}}$ and actions $\tilde{\boldsymbol{a}}$ given the augmented states $\tilde{\boldsymbol{s}}$ and the environment e. Minimizing $\mathcal{L}_{\text{robot}}$ means preserving the contacts the robot makes with the scene, which we do with inverse kinematics (Line 10 in Algorithm 2).

C. Learning the Valid Transforms Objective

As discussed above, we include a term $\mathcal{L}_{\text{valid}}$ based only on the transformation T. In some cases, such as our rope manipulation example, it may not be obvious how to define this objective manually. Our rope is very flexible, and therefore rotating the rope so that it floats in a sideways arc is invalid, but it may be valid for a stiff rope or cable. To address this, we offer a simple and data efficient algorithm for learning the transformation validity function f_{valid}.

Our method for learning f_{valid} is given in Algorithm 3. This algorithm repeatedly samples augmentations of increasing magnitude, and tests them on the system (lines 6 and 8). This generates ground truth states starting from an input state and action. The result is a dataset $\mathcal{D}_{\text{valid}}$ of examples $(T,\ y_{\text{valid}})$. We then train a small neural network $f_{\text{valid},\theta}(T)$ to predict the error y_{valid} and use the trained model as our transformation validity objective. We collect $n_{\text{valid}} = \sqrt{10^d}$ examples, where d is the dimensionality of the space of the transformation T.

This method owes its efficiency and simplicity to a few key assumptions about the system/data. First, we assume that we can collect a few (< 1000) examples from the system and test various transformations. This could be performed in a simulator, as we do in our experiments. Because the transformation validity objective is not a function of state, action, or environment, we can make simplifications to this simulation by picking states and environments which are easy to simulate. We denote this set of states and actions as Q_{valid}. Second, because the transformation parameters are low-dimensional (3 and 6 in our experiments) the trained model generalizes well with relatively few examples.

D. Application to Cluttered Planar Pushing

In this section, we describe how we apply the proposed method to learning the dynamics of pushing of 9 cylinders on a table (Figure 4). The moved object state s consists of the 2D positions and velocities of the cylinders. The robot state r is a list of joint positions, and the actions a are desired end effector positions in 2D. There is no w in this problem. The parameters T used are $SE(2)$ transforms. In this problem, any individual trajectory may include some moved cylinders and some stationary ones. In our formulation, the stationary cylinders are part of e and are not augmented, whereas the moved ones are part of s and are augmented. The robot's end effector (also a cylinder) is also augmented, and IK can be used to solve for joint configurations which match the augmented cylinders' state and preserve the contacts between the robot and the moved cylinders.

E. Application to Bimanual Rope Manipulation

In this section, we describe how we apply the proposed method to a bimanual rope manipulation problem (Figure 4). In this problem, there is a binary class label, so $w \in \{0, 1\}$, which is preserved under our augmentation (last constraint in Problem (1)). The rope is the moved object, and its state s is a set of 25 points in 3D. The robot state r is a list of the 18 joint positions, and the actions a are desired end effector positions in the robot frame. In this problem, we know that the only contacts the robot makes with the objects or environment are its grasps on the rope. Therefore, we preserve these contacts by solving for a robot state and action that match the augmented points on the rope. The parameters T used are $SE(3)$ transforms.

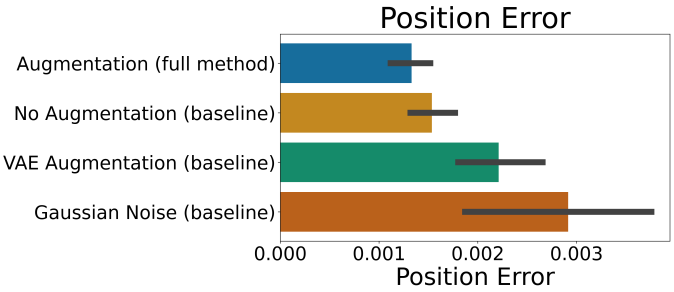

Fig. 6: Mean position error (meters) for learning the dynamics of cluttered planar pushing.

V. Results

We start by describing the tasks and our experimental methodology, then we present our results. These experiments are designed to show that training on augmentations generated by our method improves performance on a downstream task. We perform two simulated experiments, where we run thousands of evaluations, including several ablations (see Appendix 1.A). We also perform a real robot experiment (Figure 1) where we run 30 iterations of online validity classifier learning, with augmentation and without. In all experiments, we train until convergence or for a fixed number of training steps. This ensures a fair comparison to training without augmentation, despite the differing number of unique training examples. In all experiments we generate 25 augmentations per original example (See Appendix 1.B). We define key hyperparameters of our method in Appendix 1.C. A link to our code is available on the project website.

A. Cluttered Planar Pushing

The cluttered planar pushing environment consists of a single robotic arm pushing 9 cylinders around on a table. The task is to learn the dynamics, so that the motion of the cylinders can be predicted given initial conditions and a sequence of robot actions. For this, we use PropNet [10], and our task is inspired by the application of PropNet to planar pushing in [18]. The inputs to PropNet are an initial state s_0 and a sequence of actions $\boldsymbol{a}$, and the targets are the future state $(s_1, \ldots, s_T)$. All trajectories are of length 50. We evaluate the learned dynamics by computing the mean and maximum errors for position and velocity on a held-out test set. Example augmentations for this scenario are shown in Figure 3.

This is an interesting application of our augmentation for several reasons. First, it is a regression task, which few augmentation methods allow. Second, the output of the dynamics network is high-dimensional (900 for a prediction of length 50), which normally means large datasets are needed and engineering invariances into the data or network is difficult. Finally, the trajectories contain non-negligible velocities and are not quasi-static.

The original dataset contained 60 trajectories of length 50, or 3,000 time steps in total. For comparison, previous work on the same dynamics learning problem used over 100,000 time steps in their datasets [10, 18]. This is similar to the number of training examples we have *after* augmentation, which is 75,000 time steps. Finally, we measured the performance of our implementation and found that for the planar pushing scenario we generate 4.5 augmentations per second on average.

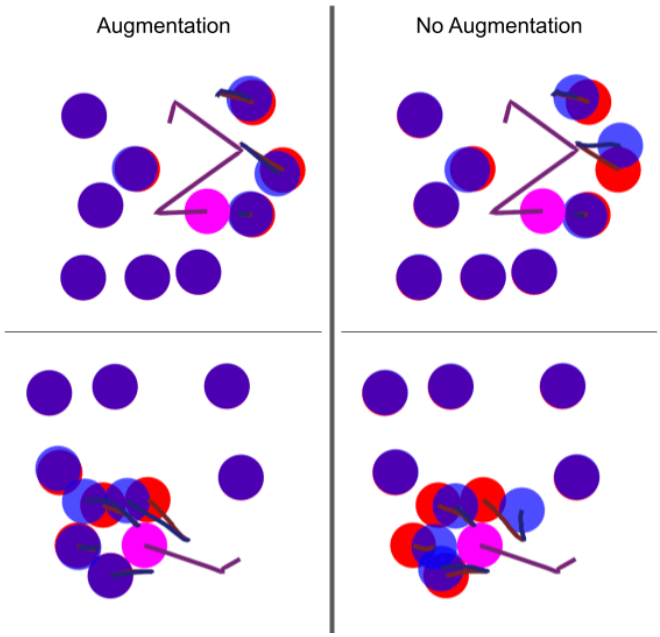

Fig. 7: Predictions (blue) vs. ground truth (red) for planar pushing. The robot is in pink. Trajectories are visualized with lines. The left column shows predictions from a model trained with augmentation, the right column without.

The primary results are shown in Figure 6. Augmentation reduces the average position error from 0.00154 m to 0.00133 m, a decrease of 14%. Additionally, we include two baselines, one which adds Gaussian noise to the state, robot, action, and environment data, and one which uses a VAE to generate augmentations as in [15]. The magnitude of the Gaussian noise was chosen manually to be small but visually noticeable. Our proposed augmentation method is statistically significantly better than the baseline without augmentation ($p < 0.0362$), the Gaussian noise baseline ($p < 0.0001$), and the VAE baseline ($p < 0.0002$). This difference in error may seem small, but note that error is averaged over objects, and most objects are stationary. Two roll-outs from with-augmentation and from without-augmentation are shown in Figure 7. In particular, we found that augmentation reduces "drift," where the model predicts small movements for objects that should be stationary. Finally, we note that the Gaussian noise and VAE baselines perform worse than no augmentation, suggesting that data augmentation can hurt performance if the augmentations are done poorly.

B. Bimanual Rope Manipulation

In this task, the end points of a rope are held by the robot in its grippers in a scene resembling the engine bay of a car, similar to [13], and shown in Figure 4. The robot has two 7-dof arms attached to a 3-dof torso with parallel-jaw grippers. The tasks the robot performs in this scene mimic

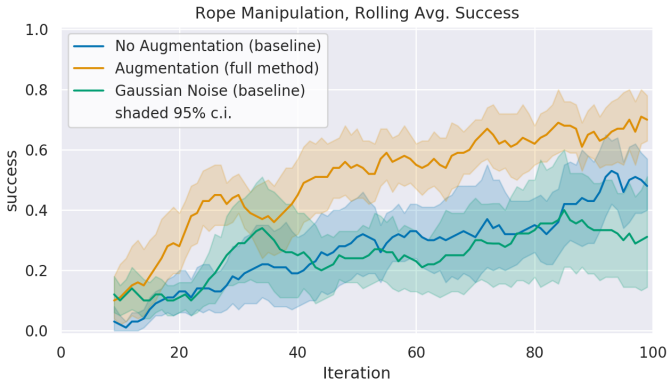

Fig. 8: The success rate on simulated bimanual rope manipulation, using a moving window average of 10.

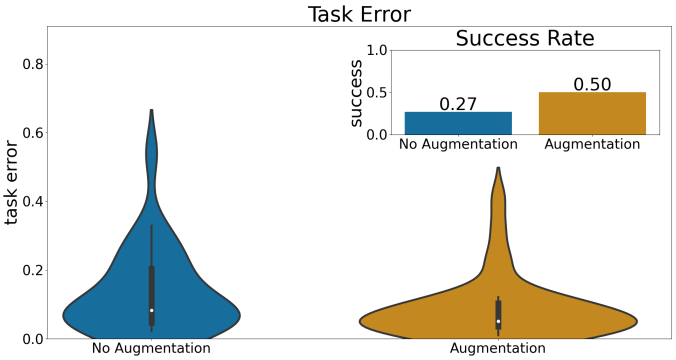

Fig. 9: The success rate and task error distribution of bimanual rope manipulation on the real robot. Task error is the distance between the goal and the final observed state of the rope.

putting on or taking off lifting straps from the car engine, installing fluid hoses, or cable harnesses. These tasks require moving the strap/hose/cable through narrow passages and around protrusions to various specified goal positions without getting caught. One iteration consists of planning to the goal, executing open-loop, then repeating planning and execution until a timeout or the goal is reached. The goal is defined as a spherical region with 4.5 cm radius, and is satisfied when any of the points on the rope are inside this region.

The planner is an RRT with a learned constraint checker for edge validity (validity classifier), and more details are given in [13]. We want to learn a classifier that takes in a single transition $x = (s_t, a_t, s_{t+1}, e_t)$ and predicts whether the transition is valid. Without a good constraint checker, the robot will plan trajectories that result in the rope being caught on obstacles or not reaching the goal. We apply our augmentation algorithm to the data for training this constraint checker. After an execution has completed, the newly-collected data along with all previously collected data are used to train the classifier until convergence. Example augmentations for this scenario are shown in Figure 2. The objective is to learn the constraint checker in as few iterations as possible, achieving a higher success rate with fewer data.

In this experiment, a total of 3,038 examples were gathered (before augmentation, averaged over the 10 repetitions). Since the purpose of our augmentations is to improve performance using small datasets, it is important that this number is small. In contrast, prior work learning a similar classifier used over 100,000 examples in their datasets [13, 12]. This is similar to the number of training examples we have *after* augmentation, which is 75,950 on average. Finally, we measured the performance of our implementation and found that for the rope scenario we generate 27 augmentations per second on average.

The primary results are shown in Figure 8. Over the course of 100 iterations, the success of our method using augmentation is higher than the baseline of not using augmentation, as well as the Gaussian noise baseline. We omit the VAE baseline, since it performed poorly in the planer pushing experiment. Furthermore, it is computationally prohibitive to retrain the VAE at each iteration, and fine-tuning the VAE online tends to get stuck in bad local minima. The shaded regions show the 95th percentile over 10 runs. If we analyze the success rates averaged over the final 10 iterations, we find that without augmentation the success rate is 48%, but with augmentation the success rate is 70%. The Gaussian noise baseline has a final success rate of 31%. A one-sided T-test confirms statistical significance ($p < 0.001$ for both).

C. Real Robot Results

In this section, we perform a similar experiment to the simulated bimanual rope manipulation experiment, but on real robot hardware. This demonstrates that our method is also effective on noisy sensor data. More importantly, it demonstrates how augmentation enables a robot to quickly learn a task in the real world. We use CDCPD2 [20] to track the rope state. The geometry of the car scene is approximated with primitive geometric shapes, like in the simulated car environment.

We ran the validity classifier learning procedure with a single start configuration and a single goal region, both with and without augmentation. After 30 iterations of learning, we stop and evaluate the learned classifiers several times. With augmentation, the robot successfully placed the rope under the engine 13/26 times. Without augmentation, it succeeded 7/26 times. The Gaussian noise and VAE baselines performs poorly in simulated experiments, therefore we omit them in the real robot experiments.

VI. Limitations

There are problems and applications where the proposed objective functions do not ensure validity, relevance, and diversity. In these cases, the structure of our augmentation and projection procedures can remain, while new objective functions are developed. Another limitation is that our method is not compatible with image data. Much recent research in robotics has moved away from engineered state representations like poses with geometric information, and so there are many learning methods which operate directly on images. Although this is a limitation of the proposed method, many of the augmentations developed for images are also not applicable to problems in manipulation, even when images are used. For

instance, pose detection, 3D reconstruction, semantic segmentation, and many other tasks may not be invariant to operations like cropping, flipping, or rotating. Creating an augmentation method for manipulation that is applicable to images is an open area for future research.

VII. Conclusion

This paper proposes a novel data augmentation method designed for trajectories of geometric state and action robot data. We introduce the idea that augmentations should be valid, relevant, and diverse, and use these to formalize data augmentation as an optimization problem. By leveraging optimization, our augmentations are not limited to simple operations like rotation and jitter. Instead, our method can find complex and precise transformations to the data that are valid, relevant, and diverse. Our results show that this method enables significantly better downstream task performance when training on small datasets. In simulated planar pushing, augmentation decreases the prediction error by 14%. In simulated bimanual rope manipulation, the success rate with augmentation is 70% compared to 47% without augmentation. We also perform the bimanual rope manipulation task in the real world, which demonstrates the effectiveness of our algorithm despite noisy sensor data. In the real world experiment, the success rate improves from 27% to 50% with the addition of augmentation.

Acknowledgments

The authors would like to acknowledge Andrea Sipos for her ingenious design of a reset mechanism, allowing us to run robot experiments unattended. This work was supported in part by NSF grants IIS-1750489 and IIS-2113401, ONR grant N00014-21-1-2118, and the Toyota Research Institute. This article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

[1] Gregory W. Benton, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. Learning Invariances in Neural Networks from Training Data. *NeurIPS*, 2020.

[2] Sumeyra Demir, Krystof Mincev, Koen Kok, and Nikolaos G. Paterakis. Data augmentation for time series regression: Applying transformations, autoencoders and adversarial networks to electricity price forecasting. *Applied Energy*, 2021.

[3] Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A Survey of Data Augmentation Approaches for NLP. *Association for Computational Linguistics*, 2021.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. *NeurIPS*, 2014.

[5] Ekin Dogus Cubuk and Barret Zoph and Dandelion Mané and Vijay Vasudevan and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *ArXiv*, 2018.

[6] Brian Kenji Iwana and Seiichi Uchida. An Empirical Survey of Data Augmentation for Time Series Classification with Neural Networks. *PLOS One*, 2020.

[7] Rico Jonschkowski and Oliver Brock. Learning state representations with robotic priors. *Autonomous Robots*, 2015.

[8] Oliver Kroemer, Scott Niekum, and George Konidaris. A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms. *JMLR*, 2021.

[9] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement Learning with Augmented Data. *NeurIPS*, 2020.

[10] Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B. Tenenbaum, Antonio Torralba, and Russ Tedrake. Propagation Networks for Model-Based Control Under Partial Observation. *ICRA*, 2019.

[11] Edward Ma. NLP Augmentation, 2019.

[12] Dale McConachie, Thomas Power, Peter Mitrano, and Dmitry Berenson. Learning When to Trust a Dynamics Model for Planning in Reduced State Spaces. *IEEE Robotics and Automation Letters*, 2020.

[13] Peter Mitrano, Dale M^{c}Conachie, and Dmitry Berenson. Learning Where to Trust Unreliable Models in an Unstructured World for Deformable Object Manipulation. *Science Robotics*, 2021.

[14] David Navarro-Alarcón, Yun-Hui Liu, José Guadalupe Romero, and Peng Li. Model-Free Visually Servoed Deformation Control of Elastic Objects by Robot Manipulators. *IEEE Transactions on Robotics*, 2013.

[15] Hiroshi Ohno. Auto-encoder-based generative models for data augmentation on regression problems. *Soft Computing*, 2020.

[16] Connor Shorten and Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6:60, 2019.

[17] P.Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied to visual document analysis. *International Conference on Document Analysis and Recognition*, 2003.

[18] Ahmet E. Tekden, Aykut Erdem, Erkut Erdem, Mert Imre, M. Yunus Seker, and Emre Ugur. Belief Regulated Dual Propagation Nets for Learning Action Effects on Groups of Articulated Objects. *ICRA*, 2020.

[19] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle J. Palmer, and Ian D. Reid. A Bayesian Data Augmentation Approach for Learning Deep Models. *NeurIPS*, 2017.

[20] Yixuan Wang, Dale McConachie, and Dmitry Berenson. Tracking Partially-Occluded Deformable Objects while Enforcing Geometric Constraints. *ICRA*, 2021.

[21] Johnny Wood. Robot workers are being hired at record rates in US companies - here's why. *World Economic Forum*, 2021.

[22] Kuan-Ting Yu, Maria Bauzá, Nima Fazeli, and Alberto Rodriguez. More than a Million Ways to Be Pushed: A High-Fidelity Experimental Data Set of Planar Pushing. *IROS*, 2016.

[23] Wojciech Zaremba. OpenAI Disbands Robotics. *Interview on YouTube*, 2021.

[24] Sheng Zhong, Zhenyuan Zhang, Nima Fazeli, and Dmitry Berenson. TAMPC: A Controller for Escaping Traps in Novel Environments. *IEEE Robotics and Automation Letters*, 2021.

[25] Gao Ziyan, Armagan Elibol, and Nak Young Chong. Planar Pushing of Unknown Objects Using a Large-Scale Simulation Dataset and Few-Shot Learning. *International Conference on Automation Science and Engineering (CASE)*, 2021.