2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Focused Adaptation of Dynamics Models for Deformable Object Manipulation

Peter Mitrano[1] and Alex LaGrassa[2] and Oliver Kroemer[2] and Dmitry Berenson[1]

***Abstract*— In order to efficiently learn a dynamics model for a task in a new environment, one can adapt a model learned in a similar source environment. However, existing adaptation methods can fail when the target dataset contains transitions where the dynamics are very different from the source environment. For example, the source environment dynamics could be of a rope manipulated in free space, whereas the target dynamics could involve collisions and deformation on obstacles. Our key insight is to improve data efficiency by focusing model adaptation on only the regions where the source and target dynamics are similar. In the rope example, adapting the free-space dynamics requires significantly less data than adapting the free-space dynamics while also learning collision dynamics. We propose a new method for adaptation that is effective in adapting to regions of similar dynamics. Additionally, we combine this adaptation method with prior work on planning with unreliable dynamics to make a method for data-efficient online adaptation, called FOCUS. We first demonstrate that the proposed adaptation method achieves statistically significantly lower prediction error in regions of similar dynamics on simulated rope manipulation and plant watering tasks. We then show on a bimanual rope manipulation task that FOCUS achieves data-efficient online learning, in simulation and in the real world.**

I. INTRODUCTION

Autonomous systems often rely on a dynamics model, which predicts future states given actions, to reach a desired goal state. However, general and accurate dynamics models only exist for a narrow range of robotics problems. Learning these dynamics models is an increasingly popular paradigm, in part because learned models can be repeatedly improved using autonomously collected real-world data. However, fine-tuning an initial dynamics model on new data can perform poorly when the data contains complex dynamics on which the dynamics model was not initially trained. For example, suppose we want to manipulate a rope amongst clutter, and we have a dynamics model trained on free-space motions in simulation. Free-space transitions in the real world are fairly similar to free-space transitions in simulation, but transitions where the rope deforms on objects in the scene are very different from anything seen in simulation. We call these transitions *distracting*, because they are hard to learn from a few examples, and because they make it harder to adapt accurately to the free-space dynamics. More generally, transitions from regions of dissimilar dynamics can inhibit effective transfer to regions of similar dynamics. This problem is similar to "cleaning" data in machine learning [1]–[3]. For dynamics learning, defining what "clean" means can be difficult, and has not been studied extensively. Instead, the dominant paradigm is simply to train on all the collected data.

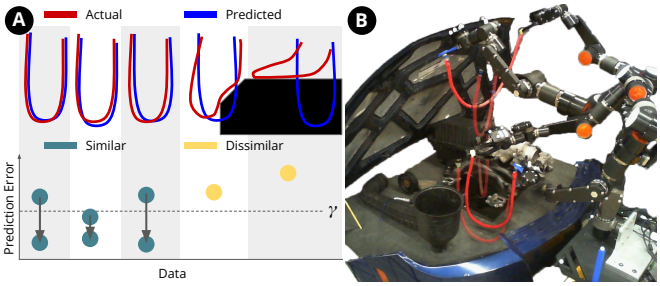

Fig. 1: (A) An illustration of how our adaptation method focuses on regions where the source and target dynamics are similar. When focusing adaption on free-space dynamics, the prediction errors decrease for other free-space data (similar), and do not decrease for collision dynamics (dissimilar). (B) 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 our proposed adaptation method to improve the success rate during online learning for this task.

However, training on all the data can fail because real-world datasets for learning dynamics are often too small to learn generalized models over the entire state-action space. In our experiments, we show that simply fine-tuning on all the data can yield a model that is not accurate enough for planning. If the task can be completed while remaining in regions where dynamics are similar, then it can be worth trading accuracy in dissimilar regions for accuracy in similar regions. Our key insight is that, when we are adapting from an initial model, we can leverage the initial model to achieve significantly lower prediction error by focusing on transitions where the source and target dynamics are the most similar. The idea that transfer is easier when the source and target data are similar is well-supported in the transfer learning literature [4], [5].

To implement this strategy, we propose an adaptation method that minimizes prediction error in regions where the source and target dynamics are similar. The proposed method minimizes prediction error in these regions by fine-tuning on an initially small set of data from these regions, and growing that subset over the course of training. This is done with a loss function inspired by curriculum learning that weights transitions according to their prediction error, assigning a higher weight to low-error transitions. Under the assumption

[1]Department of Robotics, University of Michigan
[2]Robotics Institute, Carnegie Mellon University

This work was supported in part by the Office of Naval Research Grants N00014-21-1-2118 and N00014-18-1-2775, NSF grants IIS-1956163, CMMI-1925130, IIS-1750489 and IIS-2113401, and ARL grant W911NF-18-2-0218.

that there are paths to the goal where the source and target dynamics are similar, this adaptation method can be used to achieve high task success in the target environment.

The first contribution of this paper is a method for adapting dynamics models to datasets that contain distracting transitions. We demonstrate that the proposed method is successful in filtering out distracting data and that the resulting model is more accurate in the regions of state-action space where the source and target dynamics are similar. The second contribution is a data-efficient online-learning method that pairs our adaptation method with prior work on planning with unreliable dynamics models [6], [7]. We call our combined method for online learning FOCUS. FOCUS achieves higher success rates in the low-data regime because the adapted dynamics are more accurate, which leads to finding more reliable plans.

II. RELATED WORK

Adapting dynamics models: One approach to adaptation is to use system identification to fit a global dynamics model of the target domain [8]–[11]. Alternatively, domain randomization uses random variations of conditions during training to make the model robust to that type of variance during test time [12]. Some methods go further, and iteratively refine the noise distribution by comparing simulations to rollouts executed in the real world [13], [14]. These methods require knowledge of how parameters can vary between the source and the target domains. By contrast, our approach needs no knowledge of how the system parameters such as dynamical parameters, object geometry, and kinematics may vary. Prior work has also used estimates of similarity between the source and target system to guide adaptation, for instance by selecting training data most similar to the source system [4] or selecting the most useful source domain for a given target domain [15]. Similarly, our approach focuses both training and data collection on transitions with low prediction error, which are easier to learn. Previous work avoids negative transfer of dynamics from the source domain that is harmful to performance in the target domain [4], [16]. In contrast, our work focuses on avoiding distracting dynamics in the target domain that inhibit effective transfer to dynamics where the source and target are similar.

Data Cleaning: Data Cleaning is a term for methods that aim to remove training examples that are harmful, incorrect, or unhelpful [17]. Methods such as Majority Vote Filtering or the Iterative-Partitioning Filter can remove mislabeled classification data, but they do not apply to regression problems and assume that the type of data we want to fit is the majority type [1], [2], [18]. In contrast, learning dynamics is a regression problem, and in our experiments, the data we want to adapt to may be a minority.

Curriculum learning: Transfer learning methods work best between similar source and target domains [4], [16]. To tackle larger differences, curriculum learning methods use intermediate problems through mechanisms such as pseudo-labels or task selection, which has been successful in classification, reinforcement learning, and machine translation [19]–[22]. Like curriculum learning, our proposed method tries to train on easier data first, and keep the difference between the old and new data small [23]. However, unlike in standard curriculum learning, we do not necessarily converge to training on all data.

Planning with unreliable dynamics models: If a dynamics model has a known probabilistic transition model, belief-space planning can be used [24], [25]. Since meaningful predictive uncertainty distributions are difficult to estimate for novel scenarios, some methods instead directly predict model error and use it as a constraint for planning [6], [7], [26]. Rather than avoiding uncertainty, model-based reinforcement learning methods try to reduce uncertainty by collecting new data and fine-tuning the dynamics [11], [27]. In this work, we both adapt the dynamics online and avoid inaccurate predictions, which we find results in higher task success rates with less data.

Rope Manipulation and Plant Watering: Prior model-based rope manipulation and plant watering methods optimize parameters of a simulator or neural network to match real-world dynamics, as we do here. [14] also uses a mechanism to avoid including data that is high error for a particular local model. Typically, the goal is to learn a model that is accurate for a wide range of dynamics for a task, without prioritizing dynamics that may require less data to learn [28]–[35]. Since linear deformable objects are so high-dimensional, more work explores methods to learn or adapt models for a subset of configurations [6], [26], [36].

III. METHODS

First, we formalize the dynamics adaptation problem studied in this paper. Then, we describe our method for adapting the dynamics. Finally, we describe how our adaptation method can be used in an online adaptation framework to efficiently learn a rope manipulation task.

A. Problem Statement

The problem addressed in this paper is to adapt a dynamics model trained in a source environment to data collected in a target environment, where the source and target environments have dynamics that are similar in some regions of the state-action space, but different in others. Furthermore, we consider the case where data collection is done by planning and executing paths to goals in the target environment using the learned dynamics model.

To formalize this, first consider the standard dynamics learning problem with a dataset $\mathcal{D}$ of transitions of states, actions, and next states (s, a, s'). We also assume a distance function $d(s_1, s_2)$ that returns a scalar is given, and that the state is fully observable. The true dynamics are $s' = f(s, a)$, and the learned dynamics are $\hat{s}' = \hat{f}(s, a)$. The initial dynamics $\hat{f}_0$ is the model that is pre-trained in the source environment and not yet adapted to the target environment. We define the source and target environment dynamics as similar for a transition if $d\big(f_S(s, a), f_T(s, a)\big) < \gamma$. The threshold γ should be small enough that it excludes distracting transitions, but large enough to include as much data from the target environment as possible. Let $\mathcal{D}_{ST}$ be the set of

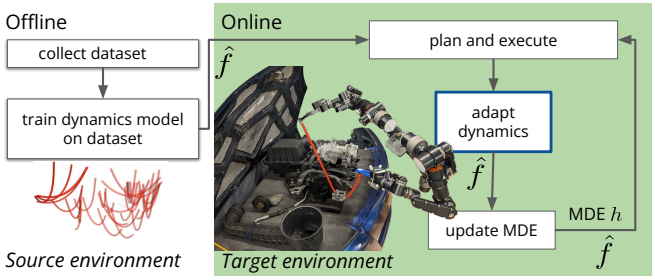

Fig. 2: Block diagram showing the steps of our full online adaptation method. A dynamics model is initialized offline in the source environment (left), then adapted online in the target environment.

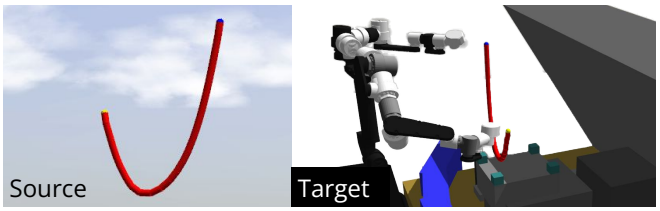

Fig. 3: Source environment for bimanual rope manipulation (left) and simulated target environment (right) where there is a robot, obstacles, and the rope damping and stiffness are changed.

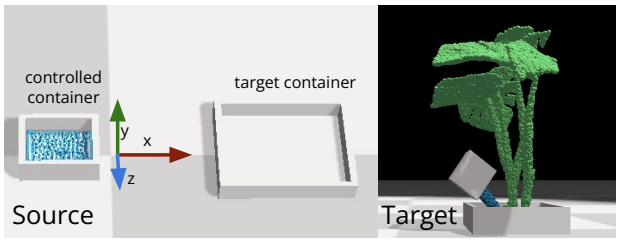

Fig. 4: Source environment for plant watering (left) and target environment (right) where there is an additional plant, and the viscosity is tripled.

transitions from the target environment where this similarity condition holds.

In order to minimize the amount of data needed for adaptation to generalize, we aim to adapt the dynamics only to transitions from regions where this similarity condition holds ($\mathcal{D}_{ST}$). However, we also care about successfully completing the task, and therefore we also assume there are paths $\{s^0, a^0, \ldots, a^{T-1}, s^T\}$ to the goal $s^T \in \mathcal{G}$ within the regions of similar dynamics $(s^t, a^t, s^{t+1}) \in \mathcal{D}_{ST}$. This also means starting states are assumed to be in $\mathcal{D}_{ST}$. If this is not the case, a different source dynamics model may be needed.

While the ultimate objective of adaptation should be to maximize task success in the target environment, an important condition for task success is minimizing prediction error on $\mathcal{D}_{ST}$. If the goal is reachable within $\mathcal{D}_{ST}$ (as we assume), the prediction error on $\mathcal{D}_{ST}$ is small (our objective), and our motion planner is constrained to stay in $\mathcal{D}_{ST}$, then we can also expect high task success. Next, we discuss how our adaptation method minimizes error on $\mathcal{D}_{ST}$, followed by how we can achieve high task success by additionally constraining a motion planner to $\mathcal{D}_{ST}$.

B. Adapting the Dynamics

Our objective is to minimize prediction error on $\mathcal{D}_{ST}$. At a high level, our method dynamically weights the training data $\mathcal{D}$ such that transitions that are likely to be in $\mathcal{D}_{ST}$ are given weights near 1, and transitions unlikely to be in $\mathcal{D}_{ST}$ are given weights near 0. Given a transition from our training data $(s, a, s') \in \mathcal{D}$, we cannot directly evaluate whether that transition is in $\mathcal{D}_{ST}$, since that would require knowing the true dynamics f_T. However, since the initial dynamics $\hat{f}_0$ is trained to have low prediction error in the *source* environment, transitions from the *target* environment which also have low prediction error $d\big(\hat{f}_0(s,a), s'\big) < \gamma$ are therefore in $\mathcal{D}_{ST}$. By training on transitions with initially low error, we expect the prediction error on other transitions which belong to $\mathcal{D}_{ST}$ to also decrease. This slowly brings more and more transitions to have prediction errors below γ. On the other hand, the prediction error is unlikely to decrease for transitions not in $\mathcal{D}_{ST}$, because they are dissimilar to the transitions with low initial error.

Thus, at each step j of training, we assign each transition a weight as a function of the prediction error $||\hat{s}' - s'||^2$ and multiply this weight by the loss. The full loss is shown in Equation (1).

$$\mathcal{L}_f = \frac{1}{T}\sum_{t=1}^{T}\Big(||\hat{s}^t - s^t||^2 w^t\Big) \qquad (1)$$
$$w^t = 1 - \sigma\big(\phi(j)(||\hat{s}^t - s^t||^2 - \gamma)\big)$$

When $\phi(j)$ is higher, the sigmoid σ makes the boundary harder. For example, we use $\phi(j) = 0.5j$ and $\phi(j) = 5j + 3$ in our experiments. When j is large, the boundary is almost hard, and transitions with error below γ have a weight near 1 and transitions with error above have a weight near 0. When j is small, the boundary is soft, and the weights vary less. We found that allowing the weighting to be soft during early training steps improves the stability in the case where few or no transitions have errors below γ at the beginning of training. Adding a constant term ensures that $\phi(j) > 0$, which starts training with a harder boundary, reducing catastrophic forgetting at the beginning. The parameter γ can be chosen based on either the maximum error that can be corrected by a low-level controller, or based on the distribution of error on a validation set from the source environment (e.g. the 97th percentile).

C. Online Learning

In this section, we describe how the proposed adaptation method can be combined with prior work on planning with unreliable dynamics to achieve data-efficient online adaptation of dynamics models. A block diagram of the full method, which we call FOCUS, is shown in Figure 2. FOCUS consists of an offline phase and an online phase. In the offline phase, we train a dynamics model using data from the source environment, which in our experiments is a simple simulation. We use random actions to collect a diverse set of data and standard techniques for training the fully-connected neural network dynamics model [6].

In the online phase, we adapt the learned dynamics model to the target environment (e.g. the real world). This process alternates between (1) collecting new data in the target

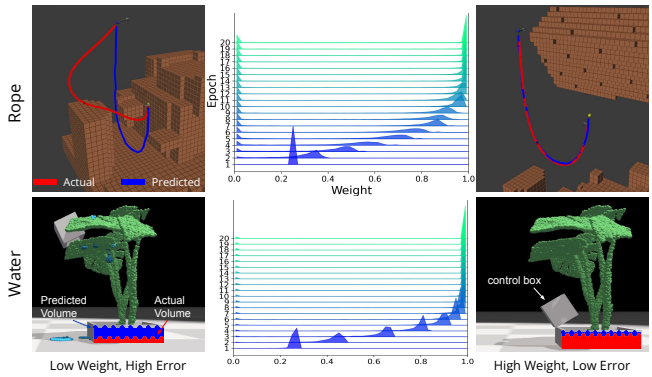

Fig. 5: (center) Histograms showing weights assigned to the data according to Equation (1) during the first 20 epochs of training. A histogram is shown for each epoch, where color varies with epoch, and these histograms are staggered along the y-axis. Initially, the weights vary only slightly across the data, but the distribution becomes strongly bimodal as training progresses. Examples of transitions given weight 0 (left) and weight 1 (right) at the end of training.

environment by planning and executing, (2) fine-tuning the dynamics, and (3) fine-tuning the model deviation estimator (MDE). We now explain the data collection and MDE fine-tuning steps.

1) Planning and Execution for Data Collection

We use a kinodynamic RRT planner where nodes are propagated using the learned dynamics model. Since the learned dynamics are adapting to only $\mathcal{D}_{ST}$ and are not accurate everywhere, we additionally constrain the planner to stay in $\mathcal{D}_{ST}$. Since $\mathcal{D}_{ST}$ is not known *a priori*, we train another neural network, called a model deviation estimator (MDE) [6], [7], [26], to predict the error of the dynamics model (more details in Section III-C.2). In addition to producing more robust plans, planning with MDEs has the additional benefit of focusing data collection on $\mathcal{D}_{ST}$. By collecting more data where the source and target dynamics are close, a larger fraction of $\mathcal{D}$ collected by the adaptation procedure is likely to be in $\mathcal{D}_{ST}$ and more useful for training.

We use the MDE in planning as a constraint. If the dynamics error predicted by the MDE is below a threshold d_{max}, then we add it to the planning tree. We also randomly accept transitions with high predicted dynamics error with low probability (0.01), so that the planner will occasionally return paths with exploratory actions (we call these *random-accepts*). These exploratory actions are essential for training the MDE since they can correct over-estimation of model error from the MDE. The threshold d_{max} for allowable error is similar to γ but may be set higher or lower to control the exploration/exploitation tradeoff.

The robot uses the planner to attempt the task, and repeatedly plans and executes open-loop until a timeout or the goal is reached. If no plan is found that reaches the goal, the plan which gets closest to the goal is executed. This repeated planning and execution is called one episode. After some fixed number of episodes (e.g. 10) we fine-tune the dynamics and the MDE using all data collected so far during the online phase.

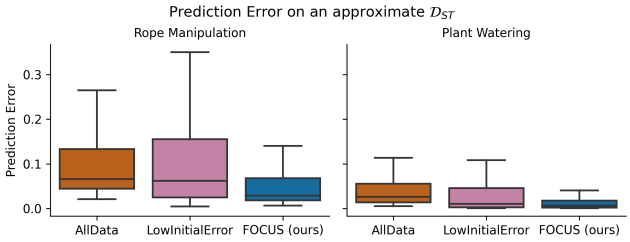

Fig. 6: Prediction error for our method versus two baselines, evaluated on a dataset of transitions from regions where the source and target dynamics are similar.

2) Fine-tune MDE

The MDE is used to constrain planning to regions where the dynamics model is predicted to be accurate, which has three benefits. First, we do not know the true error during planning, so the main purpose of the MDE is to give an estimate of error. Second, it helps bias data collection to contain transitions from $\mathcal{D}_{ST}$. Third, it makes reaching the goal more likely since it avoids plans that do not match the true dynamics. The MDE $\hat{d} = h(\mathcal{E}, s, a, \hat{s}')$ is a convolutional neural network that takes as input the environment, state, action, and next predicted state, and predicts the error of the dynamics model $\hat{d}$. We represent the environment $\mathcal{E}$ as a voxel grid of the static scene. The ground truth error used for training is the error between the true observed state and the state predicted by the learned dynamics: $d = d(\hat{s}', s')$. The loss function is shown in Equation (2). Intuitively, the MDE should be easier to learn with less data than learning the dynamics accurately everywhere, since the MDE need only predict the magnitude of the error as opposed to the full state vector [6].

$$\mathcal{L}_h = ||\hat{d} - d||^2 e^{-k_h d} \tag{2}$$

k_h is a hyperparameter that reduces the need to predict high dynamics errors with high accuracy. We set $k_h = 10$.

IV. RESULTS

We begin by describing the two domains for our experiments: bimanual rope manipulation and plant watering. We then validate the claims that (1) our proposed adaptation method achieves lower prediction error in regions of similar dynamics, and (2) that FOCUS achieves higher success rates more quickly in the online adaptation setting compared to baselines that train on all data equally.

Bimanual Rope Manipulation: In this task, a 16-DOF dual-arm robot is holding two ends of a rope in a scene resembling the engine bay of a car (scene shown in Figures 1,3). The task mimics putting on lifting straps on the engine, which requires moving the rope through narrow passages and around protrusions. The goal is to place the middle of the rope in a goal region defined as a sphere of radius $0.045\,\text{m}$. The planner outputs gripper position actions, and a local controller executes the actions while maintaining gripper orientations. The learned dynamics model predicts the state of the rope, represented as 25 points in 3D, given the initial rope state and gripper position actions. The distance function is the L2 norm of all rope points.

In the rope manipulation experiments, the source simulation has no obstacles and the robot is simplified to floating kinematic grippers. We then test adaptation to two different target environments: (1) another simulation that includes the robot and obstacles and has different damping and stiffness parameters for the rope, and (2) the real world. This tests adaptation to a different rope despite the distracting transitions where the rope deforms on the robot or the obstacles. Gazebo with ODE physics is used for simulation [37]. For rope manipulation, the set $\mathcal{D}_{ST}$ would be the transitions from the target environment where the rope is in free space. We use $\gamma = 0.08$.

Plant Watering: The goal in this task, illustrated in Figure 4 is to pour at least 75% of the initial volume from a controlled container into a target container without spilling more than 2%. The source environment is a variation from the SoftGym PourWater environment [38]. The controlled container can rotate about the z-axis. The state space is the 4-DOF $[x, y, z, \theta]$ pose of the controlled container, 3-DOF $[x, y, z]$ pose of the target container, control volume, and target volume. The action space is a target pose $[x_{des}, y_{des}, \theta_{des}]$ which is followed by a proportional controller. The distance function d is a weighted sum of the distance in x, y, theta, and volume.

The target environment has triple the viscosity, a shorter container, and a plant in the target container. Although the agent can pour from above, that causes the water to splatter, which is more difficult to predict than the free-space pours of the target environment. Additionally, the box can collide with the plant, which is dissimilar to the source dynamics where there are no obstacles. The set $\mathcal{D}_{ST}$ contains free-space motions and pours, whereas collision with and pouring on the plant is not in $\mathcal{D}_{ST}$. We use $\gamma = 0.10$.

A. Validating the Adaptation Method

We now evaluate whether the proposed adaptation method achieves lower prediction error on $\mathcal{D}_{ST}$. We start by creating validation sets that contain transitions not used for training, and which are known to be in $\mathcal{D}_{ST}$. For rope, this means transitions where the rope is in free space. For water, this means transitions that do not collide with or pour over the plant. We evaluate our method and two baselines, all starting from the same pre-trained model and adapting to the same dataset from the target environment. The baseline *AllData* fine-tunes on all transitions with equal weights. The baseline *LowInitialError* uses our weighting function, but computes the weights once using $\hat{f}_0$ and does not re-compute them throughout training. Our method uses our proposed loss function (Equation (1)) which re-computes the weights on each batch during training. We do not compare to domain randomization methods, which require knowledge of which physical parameters may vary and corresponding bounds.

For bimanual rope manipulation, the dataset contains 6288 transitions and the validation set contains 792 transitions. For plant watering, the training dataset contains 854 transitions and the validation set contains 130 transitions. The results are visualized in Figure 6. In both experiments, the error of our method is statistically significantly lower than both baselines ($p < 0.0001$).

Figure 5 demonstrates the intuition behind our adaptation method. In the center, we show histograms over time of the weights assigned to the transitions in the training dataset, for water and for rope. The distribution is initially unimodal since the weighting function when $j = 0$ is soft, but as training progresses the distribution rapidly becomes bimodal, where most transitions are given a weight of 1, but some transitions are given a weight of 0. We show examples of these low and high weight transitions on either side. For rope manipulation, we found that the number of the transitions with prediction error below γ increases from 52% at epoch 1 to 80% at epoch 20, which shows that the subset of data we train on grows. This explains why our method outperforms the *LowInitialError* baseline, since that baseline is not making use of as much of the data as our method does. The presence of transitions with 0 weight (e.g. 20% for the rope domain at epoch 20) shows that our method is not converging to training on all examples, which does not perform well based on the *AllData* baseline.

B. Online Learning Experiments

We show that FOCUS achieves higher task success with less data than baselines which fine-tune the dynamics on all available data. The first baseline, called *AllDataNoMDE* does not use our proposed adaptation method and does not use MDEs when planning, making it a conventional online learning method. The second, called *AllData* includes MDEs in planning but ablates our fine-tuning method. First, we evaluate our method in the rope manipulation domain on adaptation from one simulation to another (see Figure 3). We ran 20 iterations of online learning, where each iteration consists of 10 episodes for rope manipulation, and 27 episodes for plant watering (≈6000 total transitions during learning). We then repeated this 10 times for each method/baseline with different random seeds.

After learning, we took the models saved after each learning iteration and ran 100 episodes of evaluation per method. To maximize the success rates of all methods, we use a longer timeout and do not allow random-accepts when planning. We also stop execution and replan if the error between the plan and the observed state exceeds a large threshold on model error (0.25).

The results of this experiment are summarized in Figure 7. In both tasks, the proposed method (FOCUS) shows the highest data efficiency in achieving the goal. In the rope manipulation task, *AllData*, never finds paths to the goal. This is because its dynamics are not sufficiently accurate, and so the MDE constraint makes the planning problem infeasible. Accurately learning the dynamics using *AllData* or *AllDataNoMDE* involves predicting the deformation of the rope on obstacles, which is challenging given a dataset of only a few thousand transitions. The boundary of $\mathcal{D}_{ST}$ is less extreme in the watering task, enabling both baselines to achieve high success rates, though using more data than FOCUS. By inspecting the behavior of the methods, we found that our method quickly focuses on pours that do not interact with the plant, improving data efficiency.

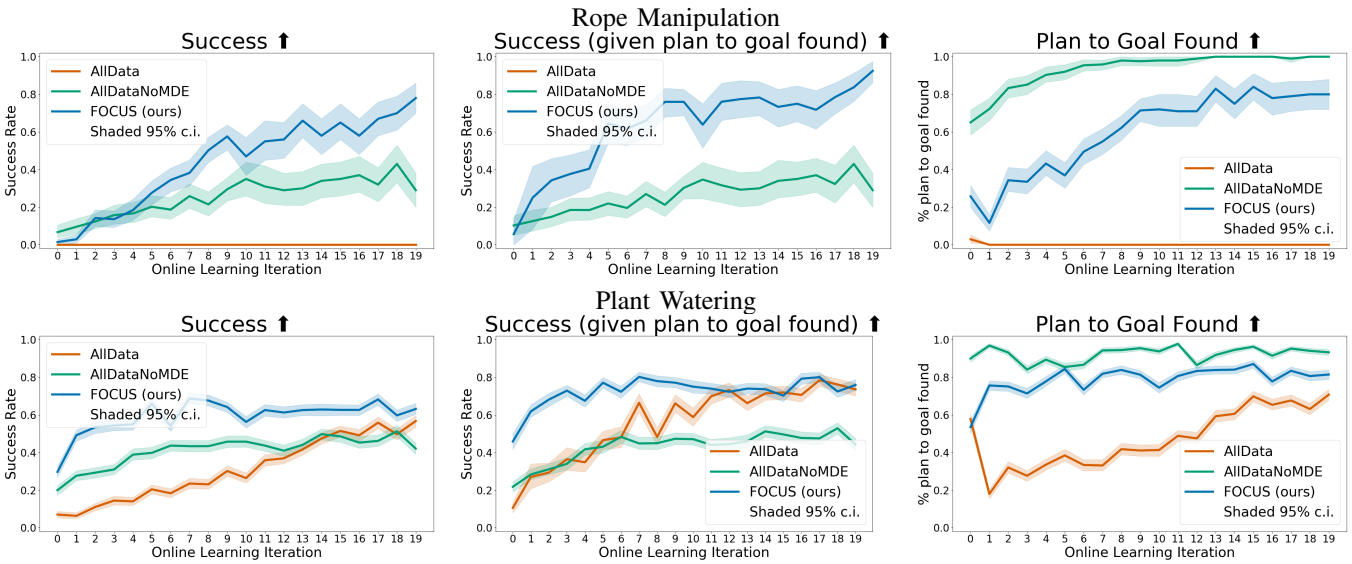

Fig. 7: Post-learning evaluation of planning during test time in simulation: Metrics shown over the 20 iterations of online learning. If the planner does not find a plan that reaches the goal, the plan with the lowest distance to the goal is executed. We report (1) overall success, (2) success given that the plan reaches the goal, (3) and the percentage of plans that reach the goal. The shaded interval is the 95% confidence interval of the mean, with the boot-strapping method used by Seaborn.

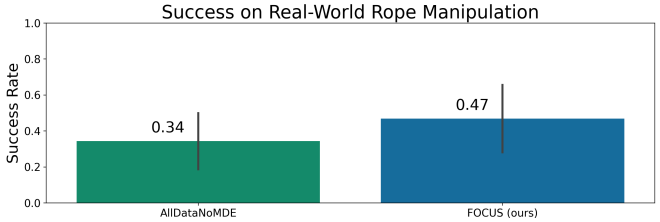

Fig. 8: Success rate for the *AllDataNoMDE* baseline (left) versus FOCUS (right) for online adaptation to real-world bimanual rope manipulation. Error bars show the 95% confidence interval.

C. Real Robot Results

To demonstrate how FOCUS enables a robot to quickly learn a task in the real world, we performed a similar experiment to the first rope manipulation experiment, but on real robot hardware. where sensor and actuator noise are substantial factors (approximately $5\,\mathrm{cm}$ of end-effector error). We use CDCPD2 [39] to track the rope state. The geometry of the car scene is approximated with primitive geometric shapes. We use the same source simulation environment as for the simulation rope experiment, but now the target environment is the real world. The robot should adapt the simulated free-space rope dynamics to the real world, despite the different real-world free-space dynamics and the fact that the rope can deform on the robots' arms or on the objects in the scene. Because perception error and actuation error are higher in the real world than in simulation, we use $\gamma = 0.2$.

We ran the online learning procedure with a single start configuration and a single goal region for one random seed and compare FOCUS to the *AllDataNoMDE* baseline, since *AllDataNoMDE* performed best in simulation. After 15 iterations of learning, we freeze the models and evaluate task success 32 times. In our unoptimized implementation, planning and fine-tuning the models took comparable amounts of time, each taking between 1 and 30 minutes depending on task complexity/dataset size. A more advanced implementation would run fine-tuning in parallel with execution so that the robot is idle only while planning.

The success rates are shown in Figure 8. With FOCUS, the robot successfully placed the rope under the engine 15/32 times, while *AllDataNoMDE* succeeded 11/32 times. Failure modes for FOCUS include the rope getting pulled out of the robots' hands, getting too close and catching on obstacles, and failing to find plans that reach the goal. We find less improvement in the real world than in simulation, which may be due to perception and significant actuator error.

V. CONCLUSION

This paper studies the problem of adapting learned dynamics models to datasets that contain transitions where the dynamics are very different from the source environment. This type of domain mismatch is common in online dynamics learning settings, where the source dynamics are learned in simulation or on a simpler task. Traditional adaptation methods can fail in this setting because trying to fit data from regions of dissimilar dynamics leads to poor predictions even in regions where the source and target dynamics are similar.

Our key insight is to instead focus adaptation on regions where the source and target dynamics are similar. We propose an adaptation method that assigns high weight to transitions with low prediction error, and dynamically re-assigns weights during the course of training. The set of low-error transitions is initially a small set, but grows as training pulls down the prediction error for other similar transitions. We combine our adaptation method with prior work on planning with unreliable dynamics to make FOCUS, a data-efficient online adaptation method. We demonstrate that FOCUS can learn a bimanual rope manipulation task in simulation and in the real world, and achieves higher task success rates than baselines that attempt to fit all the training data.

REFERENCES

[1] Carla E. Brodley and Mark A. Friedl. Identifying mislabeled training data. *Journal of Articial Intelligence Research*, 1999.

[2] Gaoxia Jiang, Wenjian Wang, Yuhua Qian, and Jiye Liang. A unified sample selection framework for output noise filtering: An error-bound perspective. *JMLR*, 2021.

[3] Daniel Bogdoll, Maximilian Nitsche, and J. Marius Zöllner. Anomaly detection in autonomous driving: A survey. *CVPR Workshop*, 2022.

[4] Michael J. Sorocky, Siqi Zhou, and Angela P. Schoellig. Experience selection using dynamics similarity for efficient multi-source transfer learning between robots. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2739–2745, 2020.

[5] Botond Bócsi, Lehel Csató, and Jan Peters. Alignment-based transfer learning for robot models. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2013.

[6] 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.

[7] Alex LaGrassa and Oliver Kroemer. Learning model preconditions for planning with multiple models. *CoRL*, 2022.

[8] Sosale Shankara Sastry and Alberto Isidori. Adaptive control of linearizable systems. *IEEE Transactions on Automatic Control*, 34(11):1123–1131, 1989.

[9] Karol Arndt, Ali Ghadirzadeh, Murtaza Hazara, and Ville Kyrki. Few-shot model-based adaptation in noisy conditions. *IEEE Robotics and Automation Letters*, 6(2):4193–4200, 2021.

[10] Ben Evans, Abitha Thankaraj, and Lerrel Pinto. Context is everything: Implicit identification for dynamics adaptation. In *2022 International Conference on Robotics and Automation, ICRA 2022, Philadelphia, PA, USA, May 23-27, 2022*, pages 2642–2648. IEEE, 2022.

[11] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[12] Kendall Lowrey, Svetoslav Kolev, Jeremy Dao, Aravind Rajeswaran, and Emanuel Todorov. Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system. In *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 35–42. IEEE, 2018.

[13] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979, 2019.

[14] Joshua D Langsfeld, Krishnanand N Kaipa, and Satyandra K Gupta. Selection of trajectory parameters for dynamic pouring tasks based on exploitation-driven updates of local metamodels. *Robotica*, 36(1):141–166, 2018.

[15] Anthony Courchesne, Andrea Censi, and Liam Paull. On assessing the usefulness of proxy domains for developing and evaluating embodied agents. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4298–4305, 2021.

[16] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[17] Xu Chu, Ihab F. Ilyas, Sanjay Krishnan, and Jiannan Wang. Data cleaning: Overview and emerging challenges. *International Conference on Management of Data*, 2016.

[18] Taghi M. Khoshgoftaar and Pierre Rebours. Improving software quality prediction by noise filtering techniques. *J. Comput. Sci. Technol.*, 2007.

[19] Ananya Kumar, Tengyu Ma, and Percy Liang. Understanding self-training for gradual domain adaptation. *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, 2020.

[20] Yang Zhang, Philip David, and Boqing Gong. Curriculum domain adaptation for semantic segmentation of urban scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 2020–2030, 2017.

[21] Runzhe Zhan, Xuebo Liu, Derek F. Wong, and Lidia S. Chao. Meta-curriculum learning for domain adaptation in neural machine translation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 14310–14318. AAAI Press, 2021.

[22] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep RL: A short survey. *IJCAI*, 2020.

[23] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. *ICML*, 2009.

[24] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013.

[25] Robert Platt Jr., Russ Tedrake, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Belief space planning assuming maximum likelihood observations. In Yoky Matsuoka, Hugh F. Durrant-Whyte, and José Neira, editors, *Robotics: Science and Systems VI, Universidad de Zaragoza, Zaragoza, Spain, June 27-30, 2010*. The MIT Press, 2010.

[26] 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.

[27] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian Processes for Data-Efficient Learning in Robotics and Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37:408–423, 2015.

[28] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. In *Conference on Robot Learning*, pages 564–574. PMLR, 2021.

[29] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4568–4575. IEEE, 2021.

[30] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*, 2018.

[31] Tatiana López Guevara, Nicholas K Taylor, Michael U Gutmann, Subramanian Ramamoorthy, and Kartic Subr. Adaptable pouring: Teaching robots not to spill using fast but approximate fluid simulation. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 2, 2017.

[32] Yoshiyuki Noda and Kazuhiko Terashima. Modeling and feedforward flow rate control of automatic pouring system with real ladle. *Journal of Robotics and Mechatronics*, 19(2):205–211, 2007.

[33] Takaaki Tsuji and Yoshiyuki Noda. High-precision pouring control using online model parameters identification in automatic pouring robot with cylindrical ladle. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2563–2568. IEEE, 2014.

[34] Zijie Li and Amir Barati Farimani. Graph neural network-accelerated lagrangian fluid simulation. *Computers & Graphics*, 103:201–211, 2022.

[35] Changhao Wang, Yuyou Zhang, Xiang Zhang, Zheng Wu, Xinghao Zhu, Shiyu Jin, Te Tang, and Masayoshi Tomizuka. Offline-online learning of deformation model for cable manipulation with graph neural networks. *RA-L*, 2022.

[36] Mingrui Yu, Kangchen Lv, Hanzhong Zhong, Shiji Song, and Xiang Li. Global model learning for large deformation control of elastic deformable linear objects: An efficient and adaptive approach. *IEEE Transactions on Robotics*, 2022.

[37] Nathan P. Koenig and Andrew Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. *IROS*, 2004.

[38] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In Jens Kober, Fabio Ramos, and Claire J. Tomlin, editors, *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pages 432–448. PMLR, 2020.

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