

MIT Open Access Articles

Visual Prediction of Priors for Articulated Object Interaction

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

Citation: Moses, Caris et al. "Visual Prediction of Priors for Articulated Object Interaction." 2020 IEEE International Conference on Robotics and Automation, May-August 2020, virtual, Institute of Electrical and Electronics Engineers, September 2020. © 2020 IEEE

As Published: <http://dx.doi.org/10.1109/icra40945.2020.9196541>

Publisher: Institute of Electrical and Electronics Engineers (IEEE)

Persistent URL: <https://hdl.handle.net/1721.1/130052>

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Terms of use: Creative Commons Attribution-Noncommercial-Share Alike



Visual Prediction of Priors for Articulated Object Interaction

Caris Moses*, Michael Noseworthy*, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Nicholas Roy

Abstract—Exploration in novel settings can be challenging without prior experience in similar domains. However, humans are able to build on prior experience quickly and efficiently. Children exhibit this behavior when playing with toys. For example, given a toy with a yellow and blue door, a child will explore with no clear objective, but once they have discovered how to open the yellow door, they will most likely be able to open the blue door much faster. Adults also exhibit this behaviour when entering new spaces such as kitchens. We develop a method, *Contextual Prior Prediction*, which provides a means of transferring knowledge between interactions in similar domains through vision. We develop agents that exhibit exploratory behavior with increasing efficiency, by learning visual features that are shared across environments, and how they correlate to actions. Our problem is formulated as a Contextual Multi-Armed Bandit where the contexts are images, and the robot has access to a parameterized action space. Given a novel object, the objective is to maximize reward with few interactions. A domain which strongly exhibits correlations between visual features and motion is kinematically constrained mechanisms. We evaluate our method on simulated prismatic and revolute joints.¹

I. INTRODUCTION

Humans frequently encounter new objects and are able to successfully articulate them with little to no experience on those particular object instances. Consider a human entering a new kitchen. They can quickly open the cupboards, turn on the lights, and operate the stove, even though they have never used these specific objects before. This behaviour is enabled by rich visual cues such as the presence of a handle, or the location of hinges on a door (see Figure 1). These features are useful for inferring the function of a new mechanism and for estimating the motion that it can undergo.

Our goal is to enable a robot to efficiently interact with novel articulated objects without human guidance by learning from previous visuo-motor experience with related objects. Previous work has shown how robots can infer the kinematic models of new mechanisms given a single demonstration of the mechanism being actuated [1]. However, demonstrations are often expensive as they require a human teacher every time the robot needs to interact with a new object. Other methods provide exploration strategies which enable a robot to estimate the kinematic properties of mechanisms [2], [3]. While these methods perform well, they do not transfer any experience from similar mechanisms when interacting



Fig. 1: *Contextual Prior Prediction* is motivated by the fact that most objects have rich visual features which indicate their motion. In our simulated door domain (top left), the position of the handle and width of the door indicate the direction which it opens and with what radius. Other objects, such as latches (top right), toys (bottom left), or ovens (bottom right) also have visible indicators such as tracks, hinges, and knobs.

with novel mechanisms. We desire a solution where the robot uses past experience to experiment efficiently with new mechanism instances to learn how to actuate them from a very small number of self-selected actuations.

Due to constraints present in articulated objects, very few of the possible motion commands the robot can generate are likely to cause the mechanism to move. Without models of the object, the robot must propose its own goals or sequences of actions that can quickly generate motion that exhibits the correct kinematic structure of the mechanism. In order to enable efficient exploration, we propose our method, *Contextual Prior Prediction* (CPP), which uses the visual appearance of a mechanism to provide a prior that indicates which actions are likely to be successful in actuating the mechanism. The mapping from visual appearance to an actuation prior is learned from previous interactions with mechanisms from the same class. This enables the robot to only try actions that it believes are likely to succeed based on the new object’s appearance.

Thus we have two learning problems. The *outer* problem is to learn a general mapping from the appearance of a mechanism and a proposed action to a *reward* which measures

*Equal contribution.

Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA
{carism, mnosew, lpk, tlp, nickroy}@mit.edu

We gratefully acknowledge the funding support of the Honda Research Institute and the Education Office at the Charles Stark Draper Laboratory.

¹Videos and code are available at <https://sites.google.com/view/contextual-prior-prediction>.

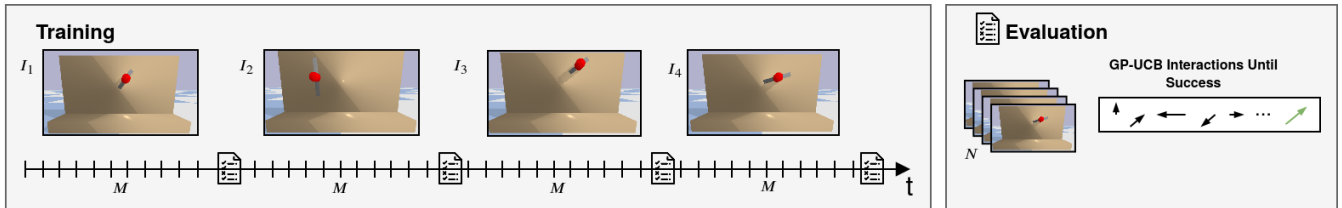


Fig. 2: In the learning setup, the robot is sequentially presented with L mechanisms. The robot can interact with each mechanism for M steps (timeline ticks) before a new one appears. After each training mechanism, we evaluate the robot’s performance on a separate evaluation set of N novel mechanisms (right). For each evaluation mechanism, the robot takes actions until it has generated an optimal interaction (generated the most possible motion).

how far the mechanism moves, i.e. a reward function. The *inner* problem is, given a particular object, and starting with a prediction based on its visual appearance, to efficiently and actively select a sequence of robot motion commands that will cause the mechanism to move. The system operates in a loop: given a new mechanism, the robot interacts with it and discovers how it moves; then, the visual appearance of that object together with each of the attempted actions and their effects are added to a training set for the *outer* learning problem. As the robot gets more experience with different mechanisms, it is able to “understand” a new mechanism with fewer and fewer trials.

We formulate the overall problem as a *Contextual Multi-Armed Bandit* (C-MAB) in which the robot continuously interacts with a sequence of mechanisms (contexts) with shared structure. For the *outer* problem, we represent the overall reward function using a neural network, which maps visual appearances and possible actions to value, and train it using conventional supervised-learning methods. Then, for the *inner* problem, given a novel mechanism to interact with, we use the neural network to predict the expected value of each possible action in a continuous action space. We treat this function as the prior mean of a *Gaussian Process* (GP), and use the *Gaussian Process Upper Confidence Bound* (GP-UCB) strategy [4] to handle the exploration-exploitation tradeoff when finding the optimal action.

In our method, GP-UCB is also used in the data collection phase. In this paper we explore the effectiveness of GP-UCB and random sampling as exploration strategies during training. At evaluation time, we compare the overall effectiveness of the system to one that applies GP-UCB to each new mechanism starting from a generic prior, as well as to a baseline random search. The techniques are generic and could apply to a variety of mechanisms. Our experimental comparisons are done in a simulated domain containing prismatic and revolute joints with different visual appearances and kinematic parameters.

The contribution of this paper is to demonstrate how a period of exploration with mechanisms with different visual appearances can lead to the ability to actuate never-before-seen mechanism instances with very few (sometimes just one) trials. In addition, the approach of learning to map appearance to the prior of a GP, rather than mapping appearance directly to motor commands, means that the overall system

is significantly more robust, and can recover from inaccurate predictions that arise when there is little training data.

II. METHOD

In this section we introduce *Contextual Prior Prediction* and discuss how CPP can be used to find optimal actions in novel contexts. Section II-A describes how CPP fits into the C-MAB framework. Section II-B reviews GP-UCB, a method which can be used to learn a context-specific reward function, and how we extend this method to C-MABs by learning a prior on the context-specific reward function. In section II-C we discuss the structure of our prior, and how it enables visual feature learning.

A. Problem Formulation

Our problem can be formulated as a *Contextual Multi-Armed Bandit* in which a robot is given a context, I , to interact with, then selects an action with the objective of maximizing its reward. We assume the actions are a continuously parameterized set of motion primitives, e.g., a pushing or twisting motion, where the parameters encode variations such as the direction of force, contact point, axis of rotation, etc. As a result, the action space is typically a continuous-valued domain such as \mathbb{R}^d or $SE(d)$. Our setup differs from the typical C-MAB in that the robot will get several interactions with each context. CPP provides a way to leverage previous experience, in the form of a prior on the context-specific reward function, when interacting with novel contexts. We evaluate our method in a learning framework in which there are separate training and evaluation phases, as shown in Figure 2.

During the training phase, the robot is sequentially given L different mechanisms and is allowed to select M interactions with each, and observe the rewards, that is, the degree to which the robot was able to actuate the mechanism. This data is used to train a model. We then carry out evaluation trials on N novel contexts: in each one, the robot is able to interact until it fully actuates the mechanism. In each evaluation context, I_n , the robot uses the learned model as a prior on the context-specific reward function to quickly maximize it. To measure the robot’s success we assume that an oracle is able to provide the optimal reward, r_n^* , and we calculate the *normalized simple regret*, e , which is the loss of not selecting the optimal action. Here, r is the reward for the robot’s chosen action.

$$e = \frac{r_n^* - r}{r_n^*} \quad (1)$$

Therefore, during each evaluation, we count the number of interactions needed until the robot can achieve regret less than a specified threshold ($e < 0.05$ in our experiments).

More formally, let \mathcal{I} be the space of possible contexts and \mathcal{A} be the space of actions that the robot can execute. We are interested in learning reward function $R : \mathcal{I} \times \mathcal{A} \rightarrow \mathbb{R}$, which specifies the value of taking an action given an image. We will learn an approximation of R , represented as a *Neural Network* (NN), denoted f_{NN} .

B. Gaussian Process Optimization with a Learned Prior

At evaluation time, if we had high confidence in the NN’s prediction of reward, we would, given a new image context I , simply execute the action

$$a^* = \arg \max_{a \in \mathcal{A}} f_{\text{NN}}(I, a) . \quad (2)$$

However, when we have a small amount of experience that prediction may not be very accurate, particularly in novel contexts. We would like to extend the model such that when the robot takes actions and fails, the reward from failed attempts inform future interactions as the robot searches for the optimal action.

The *Gaussian Process* provides one method for inferring the value of some unknown function $f \sim GP(\mu, k)$, where μ is the mean function defined on domain \mathcal{A} , f is a function on that same domain, and k is a kernel function which models the covariance between pairs of function values depending on the domain points at which the function is evaluated [5]. Given $t - 1$ samples of action and reward pairs $(a_{0:t-1}, r_{0:t-1})$, the GP provides a Gaussian posterior $\mathcal{N}(\mu_{t-1}(a), \sigma_{t-1}(a))$ over the reward of the next action a_t for any action $a_t \in \mathcal{A}$.

A simple procedure would be to take the action with the largest mean reward given the actions so far, but GP-UCB combines the mean and the variance to allow the robot to choose actions that have the potential for high reward due to a high degree of uncertainty in the model. The GP-UCB criterion [4] generates an action sample a_t , given both the prior and the values of the previous samples, according to the objective

$$a_t = \arg \max_{a \in \mathcal{A}} \mu_{t-1}(a) + \beta_t^{1/2} \sigma_{t-1}(a) \quad (3)$$

where $\mu_{t-1}(a)$ and $\sigma_{t-1}(a)$ are the mean and variance of the function value at a at time $t - 1$, and β is a parameter which trades off between sampling parameters with a high mean and those with high variance.

Typically the mean function is initialized to zero, $\mu_0(a) = 0$, but we have the pre-trained NN to provide guidance before any new data is collected. To incorporate the NN, we use the GP to model the residual function f_{RI} between the true reward function and our prediction for context I , so $f_{\text{RI}}(a) = R(I, a) - f_{\text{NN}}(I, a)$. On iteration t of the GP-UCB optimization process, we select

$$a_t = \arg \max_{a \in \mathcal{A}} f_{\text{NN}}(I, a) + \mu_{t-1}(a) + \beta_t^{1/2} \sigma_{t-1}(a). \quad (4)$$

The GP will have a prior mean function that is constant 0, and a fixed kernel function, but the above function is equivalent to putting a GP on $R(I, \cdot)$ with prior mean function $f_{\text{NN}}(I, \cdot)$. Under this criteria, samples initially come from parts of the space where the learned reward function predicts high reward. Then as the true context-specific reward function is learned, we select actions with much more accurate knowledge of the true underlying reward function.

In this framework, to select actions according to the GP-UCB criteria given a context I , we must optimize Equation 4. To do this we start by evaluating the function on a coarse sampling of the action space. We then perform non-linear optimization on a few of the best samples, and select the best optimization run as the criteria-maximizing action. To find the agent’s current best estimate of the optimal action, in order to calculate Equation 1, we follow the same procedure but with $\beta = 0$.

In our experimental results we use a *squared exponential* kernel and tune the kernel parameters by executing GP-UCB on a separate set of random mechanisms and observing the resulting exploration strategies. We aim to find a good balance between exploring areas of the input space with high uncertainty and areas with known high reward.

C. Learned Reward Function

We approximate f_{NN} with a NN which consists of independent encoders for the input channels (images and action parameters) and a regressor which uses these encodings to predict reward. The image encoder f_{im} has the form $z_{\text{im}} = f_{\text{im}}(I) = f_{\text{ss}}(f_{\text{cnn}}(I))$, where f_{cnn} is a *Convolutional Neural Network* (CNN). We found that mapping from the CNN directly into a fully connected layer did not result in useful encodings, so we added f_{ss} , which is a *spatial softmax* layer [6]. It generates, as output, a set of 2D feature points that are salient for making value predictions. Each 2D feature point is the expected pixel location of activations in one of the final CNN channels.

For the action inputs, a *Multi-Layer Perceptron* (MLP), f_{a} , transforms the action parameters into a latent space. This part of the network was designed so that additional action types could be added to the system, in which case, action parameters for all policies would be transformed into a shared space. This yields action encoding $z_{\text{a}} = f_{\text{a}}(a)$. Finally, z_{im} and z_{a} are concatenated and passed through an MLP, f_{dist} , which learns to predict how the action and image features map to a reward. The NN is composed of these encoders and the distance regressor as follows,

$$f_{\text{NN}}(I, a) = f_{\text{dist}}([f_{\text{im}}(I); f_{\text{a}}(a)]) . \quad (5)$$

If the NN is trained to effectively approximate the true reward function R then an approximately optimal policy has the form

$$\pi(I) = \arg \max_{a \in \mathcal{A}} f_{\text{NN}}(I, a) . \quad (6)$$

This formulation suffices for mechanisms that can be effectively actuated by a single relatively simple parameterized action. If we were to move to truly sequential mechanisms, such as gate latches, it would be necessary to treat the problem as a *Contextual Markov Decision Process* rather than a contextual bandit.

III. EXPERIMENTS

In this section we describe the effectiveness of CPP in the *Slider* and *Door* domains. We find that CPP is able to learn from relatively few training mechanisms to quickly operate a new mechanism. We further analyze different data generation methods and their performance. Our experiments use the *PyBullet* simulator [7].

A. Domains

1) *Sliders*: Sliders are prismatic joints that vary in length, position, and the angle of the slider’s track. Their action space, \mathcal{A}_{prism} , consists of parameterized actions that actuate prismatic joints. Prismatic joints are parameterized by the pose of the origin, $\mathbf{a} \in SE(3)$, a unit vector direction in the frame of the origin, $\mathbf{e} \in \mathbb{R}^3$, and the desired configuration, or distance to move the slider handle, $q \in \mathbb{R}$. We search the space of q and the pitch component of \mathbf{e} .

2) *Doors*: Doors are revolute joints that vary in size and the direction in which they open. Their action space, \mathcal{A}_{rev} , consists of parameterized actions that actuate revolute joints which are parameterized by the pose of center of rotation, $\mathbf{c} \in SE(3)$, the distance from the center of rotation along the x -axis to the mechanism’s handle, and $r \in \mathbb{R}$, and the desired configuration, or opening angle of the door, $q \in \mathbb{R}$. We search the space of r , q , and the pitch component of \mathbf{c} .

With *PyBullet*[7] we can generate multiple mechanisms where instances of the same joint type share visual structure. In our experiments, the robot will see a sequence of randomly generated mechanisms belonging to the same class. See Figure 2 for example sliders and Figure 1 for a door.

An action outputs a trajectory the end-effector should follow to actuate a joint with the corresponding parameters. Given the pose of the mechanism handle, and action parameters, a trajectory is generated in the mechanism’s configuration space. Then inverse-kinematics and Cartesian interpolation are used to generate a trajectory in Cartesian space. This trajectory is executed using a PD controller by applying forces to the handle. Due to the mechanism’s constraints, the applied forces do not always result in motion.

Training and evaluation are interleaved. As the robot interacts with *training* mechanisms, it is intermittently evaluated on a new set of *evaluation* mechanisms. It gets to interact with each new evaluation mechanism until it is able to maximize its reward. In all our experiments, the reward is the distance the mechanism moves.

B. Interacting with a New Mechanism

In this section we compare methods for interacting with a new mechanism. The CPP method can work with any kind of exploration strategy during training time. In Figure 3 we

show our method using two different exploration strategies during training time. We will discuss the comparison of these two methods more in Section III-C.

CPP-Random: We randomly sample from the action space to collect training interactions for each mechanism.

CPP-GP-UCB: We use the GP-UCB algorithm (see Section II-B) to collect training interaction data. With this method the agent is actively trying to maximize its reward with each mechanism.

We compare our method against two simple but sensible baseline methods for evaluation that do not try to use previous experience and visual information about the new mechanism to predict how to actuate it. Thus, for the baseline methods, each mechanism is a new problem. While we expect performance to improve *within* one trial of interaction with a mechanism, we do not expect performance to improve *across* interactions with different mechanisms.

Random: We randomly sample from the action space until the agent is able to maximize its reward.

GP-UCB: The robot uses the GP-UCB algorithm (see Section II-B) for action selection until it is able to maximize its reward.

In both the baselines and in our evaluation of CPP, we limit the agent to 100 attempts at maximizing its reward. The results of our experiments are shown in Figure 3. Each plot shows the number of steps each method took to maximize the reward on the evaluation mechanisms as a function of L , the number of mechanisms the robot interacted with during training time. The baseline methods (blue and green) have the same median for all L values because they are not leveraging previous experience, and thus cannot show improvement.

The learning-based methods (red and cyan) show significant decreases in the number of interactions required to generate a successful interaction. Not only does the median number of interactions to success decrease with L , but so do the quantiles, meaning these methods more reliably interact with novel mechanisms. The larger L values are important to us: a well-trained robot is able to actuate a mechanism without any experimentation!

One drawback of CPP is that it can be misled by a poorly trained NN. In this case, the robot will first explore areas where the NN predicts high reward even if it is wrong. The GP-UCB algorithm will eventually correct the model’s beliefs and explore other regions but this may take longer than an uninformed prior. We note that in all cases, as the NN starts performing better (after seeing more unique mechanisms), the number of required interactions decreases.

To visualize the usefulness of CPP versus just trusting our NN predictions, we compare an agent that simply selects the best action according to the NN, to one that uses our CPP method. Figure 4 shows a CPP agent which performs 10 GP-UCB interactions on top of the learned visual prior. As shown, CPP can still achieve low regret even with a poor NN prior.

To visualize how the NN prior improves over time, we show its predictions after being trained on an increasing

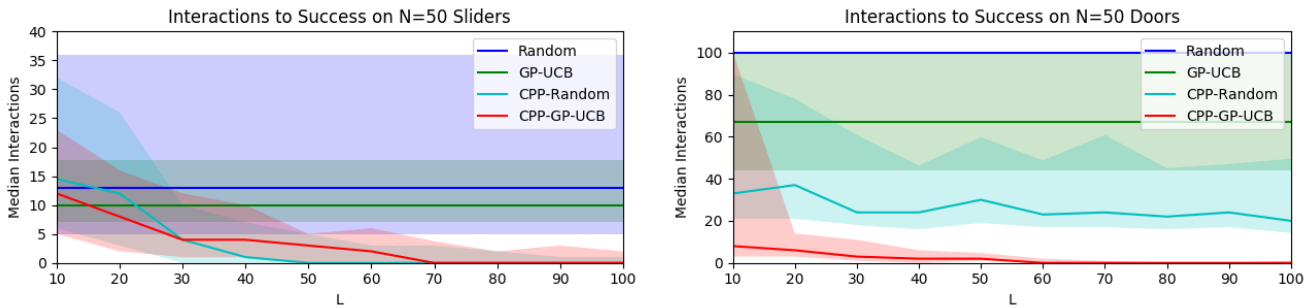


Fig. 3: Number of interactions until success (less than 0.05 regret) on N novel sliders (left) and doors (right). The median number of interactions is reported for $N = 50$ evaluation mechanisms for 5 separately trained NN models. The plots show performance for models that have been previously trained on L mechanisms (x-axes) each with $M = 100$. We compare our method, noted as CPP-GP-UCB and CPP-Random, to GP-UCB which does not learn from previous interactions, and a random baseline, Random. 25% and 75% quantiles are plotted.

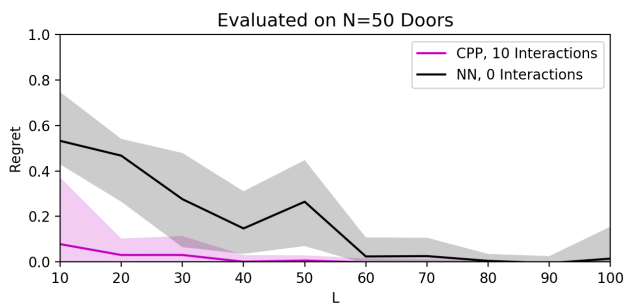


Fig. 4: This plot compares using the NN to directly predict optimal actions, to using 10 GP-UCB interactions on top of the NN to find optimal actions (CPP). As shown, the pure NN initially results in poor performance as compared to CPP.

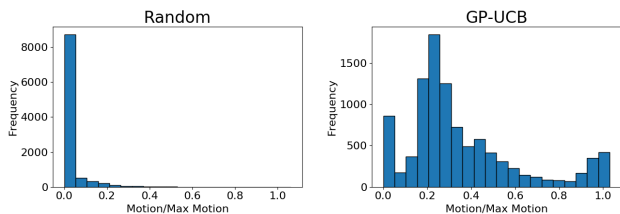


Fig. 5: Motion histograms of the data collected for $M = 100$ steps on $L = 100$ doors using random data collection (left) and GP-UCB active data collection (right). The GP-UCB sampling is biased toward collecting samples with more motion.

number of sliders in Figure 6. The predictions get better at different rates for each slider which likely correlates to how similar the evaluation sliders are to the training sliders.

C. Generation of Training Interactions

The primary utility of GP-UCB is to generate useful interactions at evaluation when the learned NN does not have accurate predictions. We also experimented with using GP-UCB for collecting useful actions while interacting with training mechanisms.

Figure 3 shows that both the CPP-Random and CPP-GP-UCB methods perform similarly in learning a good prior for slider mechanisms. However, for door mechanisms, we see

that the CPP-GP-UCB method outperforms CPP-Random. This is due to the size of the action space, the size of the rewarding region of the action space relative to the size of the entire space, and the complexity of the reward function (how dependent the policy parameters are on each other). Figure 5 gives a histogram visualization of the training data used for door mechanisms. The random exploration strategy generates mostly zero motion, while the GP-UCB method is able to effectively actively explore the action space to find rewarding samples useful for training the NN.

D. Baxter Proof of Concept

We tested our learned model for slider mechanisms on a Baxter robot. The policy parameterizations are the same, and the trajectories output by the policies are fed into a position controller for the Baxter end effector, as seen in Figure 7. Our objective was to determine if the evaluation part of our pipeline could be executed on a real robotic platform. We observed that the Baxter was able to explore the real, novel slider mechanism when it started from both a poor (little previous experience) and good NN model. The input image is a simulated version of the real mechanism as depicted in the inset of Figure 7. We surprisingly found that the Baxter was able to generate more motion for imperfect actions than the simulated agent. This was due to the compliance in the Baxter arm, which actually aided in shaping the reward, enabling the Baxter to learn the correct slider parameters with very few interactions. The robot interactions can be seen in the accompanying video.

IV. RELATED WORK

Our work lies at the intersection of estimating kinematic models and policy learning. While the former can be used in the latter, there has been little work tying the two together with the goal of generalizing to novel objects through vision.

A recent area of interest, which our work falls into, is learning low-level policies from pixel inputs. In this setting, ideally a robot could be trained to do any task with basic sensing and the right learning framework, eliminating the need to manually engineer a control system. Finn et al. [6],

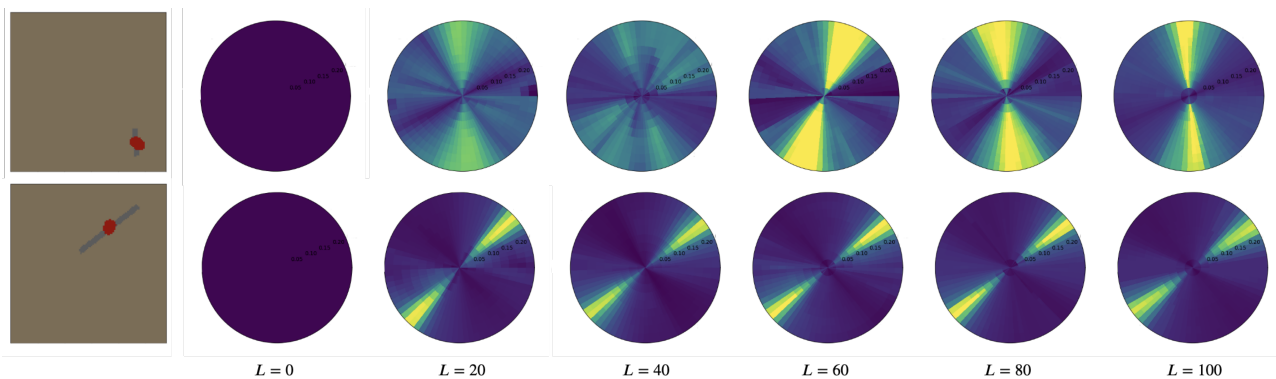


Fig. 6: Visualization of the NN prior after experience with L previous sliders for 2 novel sliders. Each plot visualizes the predicted reward for an action, given in polar coordinates (direction and distance to move the handle). Yellow indicates a higher predicted distance. In all rows the predictions improve as L increases. However they all improve at different rates. This is most likely a factor of how closely these evaluation sliders correspond to sliders seen during training.

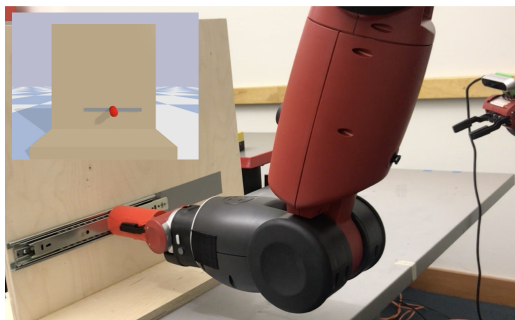


Fig. 7: The real and simulated (inset) slider actuated by the Baxter.

[8] train a network to perform various manipulation tasks from pixel-space input. Their work focuses on learning to perform a single task well, whereas we focus on manipulating multiple objects that share structure. Like Agrawal et al. [9], we use a model-based policy. Whereas they learn the forward and inverse models for poking objects, we assume we have a structured policy space based on kinematic models. Other model-based work attempts to learn a dynamics model to be used in a controller [10], [11], [12].

Recent work has started to explore the idea of learning an embedded space of policies to handle the more general case of policy learning for a variety of tasks in a single environment. In [13] they learn from teleoperated “play” data in which a human teleoperates a simulated robot to generate data to train an embedded policy space. In [14] they have a similar objective, but the data is generated via off-policy reinforcement learning. In both works the agent is constrained to perform well in only a single environment.

Another approach to learning generalizable policies is to explicitly incorporate properties that represent shared structure (e.g., dynamic, kinematic, static [15]) into the model. Of particular relevance to our work are methods that estimate the kinematic parameters of articulated objects from interaction data [1], [2], [3], [16], [17], [18]. However, these works focus on learning the kinematic parameters for a single object. In [19], the authors learn to predict the kinematic

parameters of a class of objects from an image. These works use visual representations such as optical flow [15], [16], or point features [18], which have been shown to be useful in predicting either kinematic models [16], [18] or rigid body separation [15]. Like [19], we use a CNN to learn visual features that are informative of motion in a class of objects.

A closely related problem is that of learning object affordances which is primarily concerned with the effects of actions as opposed to their rewards. In [20], the authors predict a probability distribution over possible object affordances from pixels. In [21], the authors learn the probability of successfully executing a grasp. Our method extends similar lines of work, in that the agent uses the learned model to seed on-line interactions.

GPs are useful for active exploration due to the fact that they give an estimate of the uncertainty over predictions [4]. However, using GPs requires careful specification of the kernel function which can be difficult to specify for images [22], [23]. In CPP we are able to reap the benefits of using a GP for exploration by using the image to initialize a GP over the low dimensional action space.

V. CONCLUSION

In this work, we focus on the problem of efficiently exploring novel mechanism instances by transferring knowledge from interactions with previous mechanisms through vision. We develop a method, *Contextual Prior Prediction*, that uses a NN as a prior mean for a GP which is used for exploration. We evaluate our method in a continual C-MAB learning framework on a simulated domain consisting of prismatic and revolute joints, and prove that the evaluation strategies can be executed on a real robotic platform. As the robot interacts with more mechanism instances, it can successfully actuate a new mechanism with an increasingly smaller number of interactions. Future work needs to be done evaluating what would be necessary to learn relevant features for predicting motion from realistic images. We also would like to extend our method to more complex domains which require sequential manipulation.

REFERENCES

- [1] J. Sturm, C. Stachniss, and W. Burgard, "A probabilistic framework for learning kinematic models of articulated objects," *Journal of Artificial Intelligence Research*, vol. 41, pp. 477–526, 2011.
- [2] P. R. Barragán, L. P. Kaelbling, and T. Lozano-Pérez, "Interactive bayesian identification of kinematic mechanisms," in *International Conference on Robotics and Automation (ICRA)*, 2014.
- [3] S. Otte, J. Kulick, M. Toussaint, and O. Brock, "Entropy-based strategies for physical exploration of the environment's degrees of freedom," in *International Conference on Intelligent Robots and Systems (IROS)*, 2014.
- [4] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *International Conference of Machine Learning (ICML)*, 2010.
- [5] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.
- [6] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Learning visual feature spaces for robotic manipulation with deep spatial autoencoders," in *International Conference on Robotics and Automation (ICRA)*, 2016.
- [7] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," <http://pybullet.org>, 2016–2019.
- [8] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.
- [9] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.
- [10] I. Lenz, R. A. Knepper, and A. Saxena, "DeepMPC: Learning deep latent features for model predictive control," in *Robotics: Science and Systems (RSS)*, 2015.
- [11] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik, "Learning visual predictive models of physics for playing billiards," *International Conference on Learning Representations (ICLR)*, 2016.
- [12] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *International Conference on Robotics and Automation (ICRA)*, 2018.
- [13] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *Conference on Robot Learning (CoRL)*, 2019.
- [14] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller, "Learning an embedding space for transferable robot skills," in *International Conference on Learning Representations (ICLR)*, 2018.
- [15] N. Bergström, C. H. Ek, M. Björkman, and D. Kragic, "Scene understanding through autonomous interactive perception," in *International Conference on Computer Vision Systems (ICVS)*, 2011.
- [16] D. Katz and O. Brock, "Manipulating articulated objects with interactive perception," in *International Conference on Robotics and Automation (ICRA)*, 2008.
- [17] K. Hausman, S. Niekum, S. Osentoski, and G. S. Sukhatme, "Active articulation model estimation through interactive perception," in *International Conference on Robotics and Automation (ICRA)*, 2015.
- [18] C. Eppner, R. Martín-Martín, and O. Brock, "Physics-based selection of actions that maximize motion for interactive perception," in *RSS WS: Revisiting Contact - Turning a Problem into a Solution*, 2017.
- [19] B. Abbatematteo, S. Tellex, and G. Konidaris, "Learning to generalize kinematic models to novel objects," in *Conference on Robot Learning (CoRL)*, 2019.
- [20] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Detecting object affordances with convolutional neural networks," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2765–2770.
- [21] L. Montesano and M. Lopes, "Learning grasping affordances from local visual descriptors," in *2009 IEEE 8th International Conference on Development and Learning*. IEEE, 2009, pp. 1–6.
- [22] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Gaussian processes for object categorization," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 169–188, 2010.
- [23] M. Van der Wilk, C. E. Rasmussen, and J. Hensman, "Convolutional gaussian processes," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.