2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Guided Uncertainty-Aware Policy Optimization: Combining Learning and Model-Based Strategies for Sample-Efficient Policy Learning

Michelle A. Lee*[1,2], Carlos Florensa*[1,3], Jonathan Tremblay[1], Nathan Ratliff[1], Animesh Garg[1,4], Fabio Ramos[1,5], Dieter Fox[1,6]

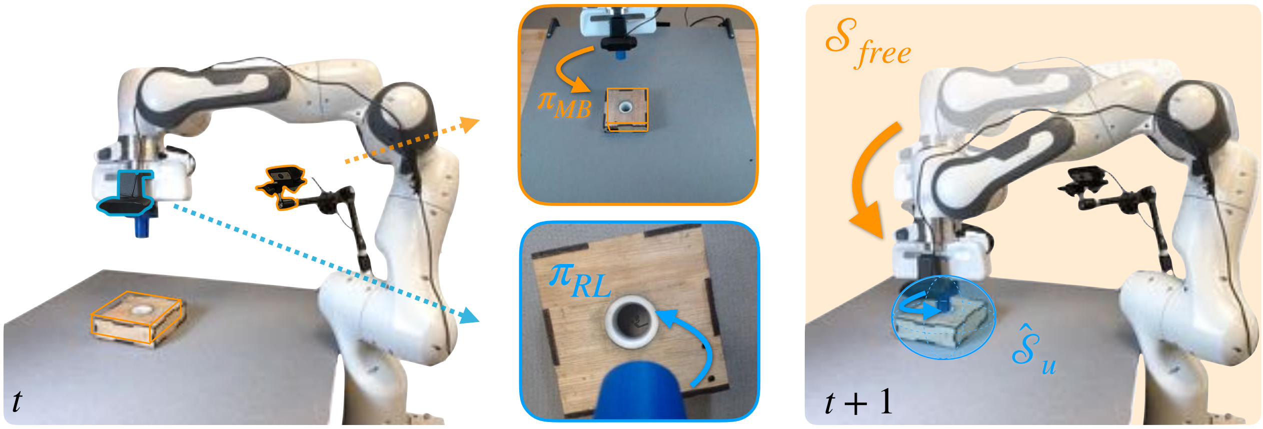


Fig. 1 Real-world setup for peg insertion: one perception system (in orange) gives the approximate position of the relevant objects. The *model-based* method, π_{MB}, drives the system from free space, $\mathcal{S}_{free}$, to the uncertainty area, $\hat{\mathcal{S}}_u$ (in blue). Once inside this area, the model can't be trusted, and a *reinforcement learning* policy, π_{RL}, is then learned directly from the raw sensory inputs of another "local" camera (in blue) that gives enough information to complete the task.

***Abstract*— Traditional robotic approaches rely on an accurate model of the environment, a detailed description of how to perform the task, and a robust perception system to keep track of the current state. On the other hand, reinforcement learning approaches can operate directly from raw sensory inputs with only a reward signal to describe the task, but are extremely sample-inefficient and brittle. In this work, we combine the strengths of model-based methods with the flexibility of learning-based methods to obtain a general method that is able to overcome inaccuracies in the robotics perception/actuation pipeline, while requiring minimal interactions with the environment. This is achieved by leveraging uncertainty estimates to divide the space in regions where the given model-based policy is reliable, and regions where it may have flaws or not be well defined. In these uncertain regions, we show that a locally learned-policy can be used directly with raw sensory inputs. We test our algorithm, Guided Uncertainty-Aware Policy Optimization (GUAPO), on a real-world robot performing peg insertion. Videos are available at: `https://sites.google.com/view/guapo-rl`.**

I. INTRODUCTION

Modern robots rely on extensive systems to accomplish a given task, such as a perception module to monitor the state of the world [1], [2], [3]. Simple perception failure in this context is catastrophic for the robot, since its motion generator relies on it. Moreover, classic motion generators are quite rigid in how they accomplish a task, *e.g.*, the robot has to pick an object in a specific way and might not recover if the grasp fails. These problems make robotics systems unstable, and hard to scale to new domains. In order to expand robotics reach we need more robust, adaptive, and flexible systems.

Learning-based method, such as Reinforcement Learning (RL) has the capacity to adapt, and deal directly with raw sensory inputs, which are not subject to estimation errors [4], [5]. The strength of RL stems from its capacity to define a task at a higher level through a reward function indicating *what* to do, not through an explicit set of control actions describing *how* the task should be performed. RL does not need specific physical modelling as they implicitly learn a data-driven model from interacting with the environment [6], allowing the method to be deployed in different settings. These characteristics are desired but come with different limitations: 1) randomly interacting with an environment can be quite unsafe for the human users as well as for the equipment, 2) RL is not recognized for being sample efficient. As such, introducing RL to a new environment can be time consuming and difficult.

Classic robotic approaches have mastered generating movements within free space, where there are no contacts with other elements in the environment [7]. We refer to these accessible methods as Model Based (MB) methods. One of their main limitations is that they normally do not handle perception errors and physical interactions naturally, *e.g.*, grasping an object, placing an object, object insertion, *etc.* As such this limits the expressiveness of roboticists and the reliability of the system.

In this work we present an algorithmic framework that is

* Equal Contribution. Work done during an internship at Nvidia.

[1] Nvidia, [2] Stanford University, [3] University of California, Berkeley, [4] University of Toronto, [5] University of Sydney, [6] University of Washington

978-1-7281-7395-5/20/$31.00 ©2020 IEEE

aware of its own uncertainty in the perception and actuation system. As such a MB guides the agent to the relevant region, hence reducing the area where the RL policy needs to be optimized and making it more invariant to the absolute goal location. Our novel algorithm combines the strengths from MB and RL. We leverage the efficiency of MB to move in free-space, and the capacity of RL to learn from its environment from a loosely defined goal. In order to efficiently fuse MB and RL, we introduce a perception system that provides uncertainty estimates of the region where contacts might occur. This uncertainty is used to determine the region where the MB method shouldn't be applied, and an RL policy should be learned instead. Therefore, we call our algorithm Guided Uncertainty Aware Policy Optimization (GUAPO).

Figure 1 shows an overview of our system, the task is initialized with MB where it guides the robot within the range of the uncertainties of the object of interest, *e.g.*, the box where to insert the peg. Once we have reached that region, we switch to RL to complete the task. At learning time, we leverage information from task completion by the RL policy to reduce our perception system's uncertainties. This work makes the following contributions:

- We demonstrate that GUAPO outperforms pure RL, pure MB, as well as a Residual policy baseline [8], [9] that combines MB and RL for peg insertion;
- We present a simple and yet efficient way to express pose uncertainties for a keypoint based pose estimator;
- We show that our approach is sample efficient for learning methods on real-world robots.

II. DEFINITIONS AND FORMULATION

We tackle the problem of learning to perform an operation, unknown *a-priori*, in an area of which we only have an estimated location and no accurate model. We formalize this problem as a Markov Decision Process (MDP), where we want to find a policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_+$ that is a probability distribution over actions $a \in \mathcal{A}$ conditioned on a given state $s \in \mathcal{S}$. We optimize this policy to maximize the expected return $\sum_{t=0}^{H} \gamma^t r(s_t, a_t)$, where $r : \mathcal{S} \rightarrow \mathbb{R}$ is a given reward function, H is the horizon of each rollout, and γ is the discount factor. The first assumption we leverage in this work can be expressed as having a partial knowledge of the transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}_+$ dictating the probability over next states when executing a certain action in the current state. Specifically, we assume this transition function is available only within a sub-space of the state-space $\mathcal{S}_{free} \subset \mathcal{S}$. This is a common case in robotics, where it is perfectly known how the robot moves while it is in free-space, but there are no reliable and accurate models of general contacts and interactions with its surrounding [10]. This partial model can be combined with well established methods able to plan and execute trajectories that traverse $\mathcal{S}_{free}$ [11], [12], [13], but these methods cannot successfully complete tasks that require acting in $\mathcal{S}_u = \mathcal{S} \setminus \mathcal{S}_{free}$. It is usually easy for the user to specify this split relative to the objects in the scene, *e.g.*, all points around or in contact with an object are not free space. If we call a point of interest, g, in that region we can therefore express that region relative to it as $\mathcal{S}_u(g)$. We do not assume perfect knowledge of the absolute coordinates of that point g nor of $\mathcal{S}_u$, but rather only a noisy estimate of them as described in Section III-A.

The tasks we consider consist on reaching a particular state or configuration through interaction with the environment, like peg-insertion, switch-toggling, or grasping. These tasks are conveniently defined by a binary reward function $r(s) = \mathbb{1}[s \in \mathcal{S}_g]$ that indicates having successfully reached a goal set $\mathcal{S}_g \subset \mathcal{S}_u$, usually described with respect to a point g [14], [15], [16]. Unfortunately this reward is extremely sparse, and random actions can take an prohibitive amount of samples to discover it [17], [18]. Therefore this paper addresses how to leverage the partial model described above to efficiently learn to solve the full task through interactions with the environment, overcoming an imperfect perception system and dynamics.

III. METHOD

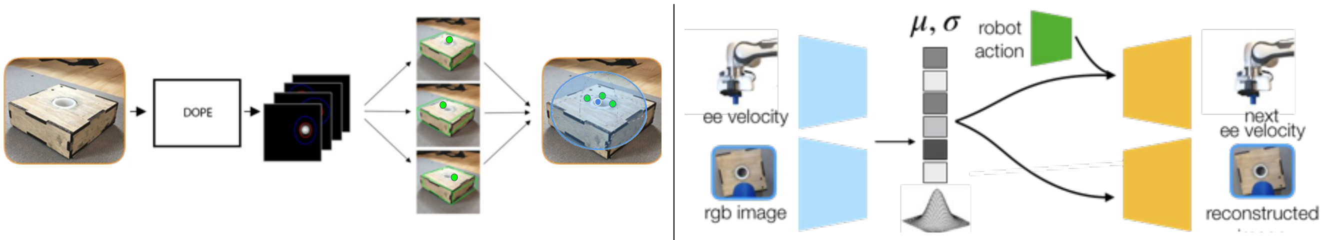
In this section, we describe the different components of GUAPO. We define $\hat{\mathcal{S}}_u$ as a super-set of $\mathcal{S}_u$ generated from the perception system uncertainty estimation. We use this set to partition the space into the regions where the MB method is used, and regions where the RL policy is trained. Then we describe a MB method that can now confidently be used outside of $\hat{\mathcal{S}}_u$ to bring the robot within that set. Finally we define the RL policy, and how the learning can be more efficient by making its inputs local. We also outline our algorithm in Algorithm 1.

A. From coarse perception to the RL workspace

Coarse perception systems are usually cheaper and faster to setup because they might require simpler hardware like RGB cameras, and can be used out-of-the-box without excessive tuning and calibration efforts [19]. If we use such a system to directly localize $\mathcal{S}_u$, the perception errors might misleadingly indicate that a certain area belongs to $\mathcal{S}_{free}$, hence trying to apply the MB method and potentially not being able to learn how to recover from there. Instead, we propose to use a perception system that also gives an *uncertainty estimate*. Many methods can represent the uncertainty by a nonparametric distribution, with n possible descriptions of the region $\{\mathcal{S}_u^i\}_{i=1}^n$ and their associated weights $p(\mathcal{S}_u^i)$. By interpreting these weights as the likelihoods $P(\mathcal{S}_u^i = \mathcal{S}_u)$, we can express the likelihood of a certain state s belonging to $\mathcal{S}_u$ as:

$$p(s \in \mathcal{S}_u) = \sum_{i=1}^{n} \mathbb{1}\big[s \in \mathcal{S}_u^i\big] p(\mathcal{S}_u^i). \quad (1)$$

If the perception system provides a parametric distribution, the above probability can be computed by integration, or approximated in a way such that the set $\hat{\mathcal{S}}_u = \{s : p(s \in \mathcal{S}_u) > \epsilon\}$ is a super-set of $\mathcal{S}_u$ for an appropriate ϵ set by the user. A more accurate perception system would make $\hat{\mathcal{S}}_u$ a tighter super-set of $\mathcal{S}_u$. Now that we have an over-approximation of the area where we cannot use our model-based method, we define a function $\alpha(s) = \mathbb{1}[s \in \hat{\mathcal{S}}_u]$



(a) DOPE perception and uncertainty to estimate $\hat{\mathcal{S}}_u$

(b) Variational autoencoder for π_{RL}

Fig. 3: Perception modules for the model-based component (left) and reinforcement learning component (right).

indicating when to apply an *RL* policy $\pi_{RL}(a|s)$ instead of the given *MB* one π_{MB}. In short, GUAPO uses a hybrid policy presented in 2.

$$\pi(a|s) = (1-\alpha(s)) \cdot \pi_{MB}(a|s) + \alpha(s) \cdot \pi_{RL}(a|s), \qquad (2)$$

Therefore we use a switch between these two policies, based on the uncertainty estimate. A lower perception uncertainty reduces the area where the *reinforcement learning* method is required, and improves the overall efficiency. We now detail how each of these policies is obtained.

B. Model-based actuation

In the previous section we defined $\hat{\mathcal{S}}_u$, the region containing the goal set $\mathcal{S}_g$ and hence the agent's reward. In our problem statement we assume that outside that region, the environment model is well known, and therefore it is amenable to use a *model-based* approach. Therefore, whenever we are outside of $\hat{\mathcal{S}}_u$, the MB approach corrects any deviations.

Our formulation can be extended for obstacle avoidance. Using a similar approach used to over-estimate the set $\hat{\mathcal{S}}_u$, we can over-estimate the obstacle set to be avoided by $\hat{\mathcal{S}}_u^{obst}$, and remove that space from where the MB method can be applied, $\mathcal{S}_{free}$. An obstacle-avoiding MB method can be used to get to the area where the goal is, while avoiding the regions where the obstacle might be, as shown in our videos[1].

C. From Model-Based to Reinforcement learning

Once π_{MB} has brought the system within $\hat{\mathcal{S}}_u$, the control is handed-over to π_{RL} as expressed in Eq. 2. Note that our switching definition goes both ways, and therefore if π_{RL} takes exploratory actions that move it outside of $\hat{\mathcal{S}}_u$, the MB method will act again to funnel the state to the area of interest. This also provides a framework for safe learning [20] in case there are obstacles to avoid as introduced in the section above. There are several advantages to having a more restricted area where the RL policy needs to learn how to act: first the exploration becomes easier, second, the policy can be local. In particular, we only feed to π_{RL} the images from a wrist-mounted camera and its current velocities, as depicted in Fig. 3b. Not using global information from our perception system in Fig. 3a can make our RL policy generalize better across locations of $\hat{\mathcal{S}}_u$. Finally, we propose to use an off-policy RL algorithm, so all the observed transitions can be added in the replay buffer, no matter if they come from π_{MB} or π_{RL}.

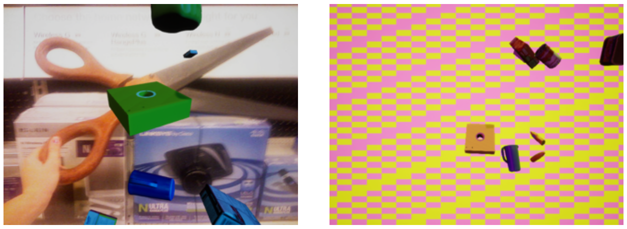


Fig. 4: Representative synthetic training images for our hole box.

D. Closing the MB-RL loop

This framework also allows to use any newly acquired experience to reduce $\hat{\mathcal{S}}_u$ such that successive rollouts can use the *model-based* method in larger areas of the state-space. For example, in the peg-insertion task, once the reward of fully inserting the peg is received, the location of the opening is immediately known. Since we no longer need to rely on the noisy perception system to estimate the location of the hole, we can update $\hat{\mathcal{S}}_u = \mathcal{S}_u$, where now the reinforcement learning algorithm only needs to do the actual insertion and not keep looking for the opening.

IV. IMPLEMENTATION DETAILS

Here we describe the implementation details of our GUAPO algorithm for a peg insertion task with a Franka Panda robot (7-DoF torque-controlled robot). We first introduce the perception module and how an uncertainty estimate is obtained to localize $\hat{\mathcal{S}}_u$. Then we describe the model-based policy used to navigate in $\mathcal{S}_{free}$ while avoiding obstacles, and the RL algorithm and the architecture of the RL policy being learned. Finally, we introduce our task set-up, the baseline algorithms we compare GUAPO with, and their implementations.

A. Perception Module

We use Deep Object Pose Estimator (DOPE) [19] as the base for our perception system. DOPE uses a simple neural network architecture that can be quickly trained with synthetic data and domain randomization using NDDS [21]. Figure 4 shows generated images with domain randomization used to train our perception system and thus allowing domain transfer (from synthetic to real world). Note that the model of the object that DOPE needs to detect is not very detailed, consisting of the approximate shape without texture. This is a

[1] https://sites.google.com/view/guapo-rl

Algorithm 1: GUAPO

```
Input: $s_0 \in \mathcal{S}$: reset state
$o^g$: global observation → RGB workspace camera
$o^l$: local observations → wrist-mounted camera, robot velocity observations
$\mathcal{S}_u$: model uncertain region containing the goal region $\mathcal{S}_g$, both unknown a-priori until the reward is obtained.
Output: $a_t$: robot actions

goal_localized ← False
for each episode do
    $s_{t=0} \leftarrow s_0$ ;                          // Reset robot
    if not goal_localized then
        $\hat{\mathcal{S}}_u \leftarrow$ DOPE($o^g$) ;      // Run perception
    end
    for $t = 0 \ldots H$ do
        if robot not in $\hat{\mathcal{S}}_u$ then
            $a_t \leftarrow \pi_{MB}$ ;   // Takes robot to $\hat{\mathcal{S}}_u$
        end
        else
            $a_t \leftarrow \pi_{RL}(o_t^l)$ ;
        end
        Apply action $a_t$ to environment ;
        $r_t = \mathbb{1}[s_t \in \mathcal{S}_g]$
        Add transition $(o_t^l, a_t, o_{t+1}^l, r_t)$ to replay buffer ;
        if $r_t$ = 1 then
            $\hat{\mathcal{S}}_u \leftarrow \mathcal{S}_u(s_t)$ ; // Lift DOPE uncert.
            goal_localized ← True ;
        end
    end
end
```

challenging case, specially because no depth sensing is used to supplement the RGB information. DOPE algorithm first finds the object cuboid keypoints using local peaks on the map. Using the cuboid real dimensions, camera intrinsics, and the keypoint locations, DOPE runs a PnP algorithm [22] to find the final object pose in the camera frame.

For this work we extended the DOPE perception system to obtain uncertainty estimates of the object pose. This extension augments the peak estimation algorithm by fitting a 2d Gaussian around each found peak, as depicted by the dark contour maps in Fig. 3a. We then run PnP algorithm on n set of keypoints, where each set of keypoint is constructed by sampling from all the 2d Gaussians. This provides n possible poses of the object consistent with the detection algorithm, as drawn in green bounding boxes in Fig. 3a. In this work we treat them as equally likely.

From our problem formulation, we assume access to a rough description of the area of interest, $\mathcal{S}_u$, around the object where an operation needs to be performed. In our peg insertion task, this is a rectangle centered at the opening of the hole. For each of the n pose samples given by our extended DOPE perception algorithm, we compute the associated hole opening positions, $\{h_i\}_{i=1}^n$, represented by the green dots in Fig. 3a. These points are then fitted by 3d Gaussian with diagonal covariance, represented in blue in the same figure. We use the mean $\hat{\mu}_{hole} = \frac{1}{n}\sum_{i=1}^{n} h_i$ as the center of $\hat{\mathcal{S}}_u$ and we over-approximate Eqn. 1 by displacing $\mathcal{S}_u$ along the axis by one standard deviation.

The perception module setup is depicted in Fig. 1 in orange, where the camera for DOPE (640x480x3 RGB images from Logitech Carl Zeiss Tessar) is mounted overlooking our workspace. The top center image with the orange border is a sample from that camera.

B. Model-Based Controller Design

As model-based controller, we use a target attractors defined by Riemannian Motion Policies (RMPs) [7] to move the robot towards a desired end-effector location. The RMPs take in a desired end-effector position $\mathbf{x} \in \mathbb{R}^3$ in Cartesian space. The target is set to be the centroid of $\hat{\mathcal{S}}_u$, which in our case corresponds to the opening of the hole $\hat{\mu}_{hole}$. As explained in the previous section, a coarse model of the object is required to train a perception module able to provide this location estimate and its uncertainty. The RMPs also require a model of the robot. These two requirements are the ones that give this part of the method the "model-based" component. By utilizing the RL component described in the following section, our GUAPO algorithm does not need these models to be extremely accurate. In case obstacles need to be avoided to reach $\hat{\mathcal{S}}_u$, we can define barrier-type RMPs.

The policies are sending end-effector position commands at 20 *Hz*. The RMPs are computing desired joint positions $\mathbf{q}_d$ at 1000 *Hz*. Given that impedance-end-effector control is an action space which has been shown to improve sample efficiency for policy learning for RL [23], we also use the RMPs interface as our reinforcement learning action space.

C. Reinforcement Learning Algorithm and Architecture

We use a state-of-the-art model-free off-policy RL algorithm, Soft Actor Critic [24]. The RL policy acts directly from raw sensory inputs. This consists on joint velocities and images from a wrist-mounted camera (64x64x3 RGB images from a Logitech Carl Zeiss Tessar) on the robot (see Fig. 1). As illustrated in Fig. 3b, all inputs are fed into a β-VAE [25]. The VAE gives us a low-dimensional latent-space representation of the state, which has been shown to improve sample efficiency of RL algorithms [26]. The parameters of this VAE are trained before-hand on a data-set collected off-line. The only part that is learned by the RL algorithm is a 2-layer MLP that takes as input the 64-dimensional latent representation given by the VAE, and produces 3D position displacement $\Delta\mathbf{x}$ of the robot end-effector.

D. Training Details

The VAE is pre-trained with 160,000 datapoints for 12 epochs, on the Titan XP GPU. DOPE is trained for 8 hours on 4 p100 GPU. All our learning-based policy methods (GUAPO, SAC baseline, and the Residual Policy baseline described in Sec. V) were trained for 60 training iterations. In total, each policy was trained with 120 training episodes, as each iteration has two training episodes, each with 1000 steps. This takes 90 min. to train.

E. Rewards

For GUAPO, we use a sparse reward when the policy finishes the task (inserts the peg). The policy gets -1 everywhere, and 0 when it finishes the task. For our other learning-based baselines (SAC [24] and Residual policy [8], [9]), we use a negative L2 norm to the perception estimate of the goal location $\hat{\mu}_{hole}$, 0 when it reaches $\hat{\mathcal{S}}_u$, and 1 when it finishes the task.

V. EXPERIMENTAL DESIGN AND RESULTS

In this section we seek to answer the following questions: How does our method compares to our baseline policies, such as, Residual policies, in terms of sample efficiency and task completion? And, is the proposed algorithm capable of performing peg insertion on a real robot?

A. Comparison Methods

All the different baselines were initialized about 75 cm away from the goal. They were all implemented on our real robotics system. As such we compare our proposed method to the following:

- **MB-Perfect.** This method consists of a scripted policy under perfect state estimation.
- **MB-Rand-Perfect.** This method uses the same policy as MB-Perfect where we injected random actions, which we sample from a normal distribution with 0 mean and a standard deviation defined by the perception uncertainty from DOPE (which is around 2.5cm to 3 cm).
- **MB-DOPE.** This method is similar to MB-Perfect, but instead uses the pose estimator prediction to servo to the hole and accomplish insertion.
- **MB-Rand-Dope.** This method uses the same policy as MB-Dope where we injected random actions, which is sampled in the same way as MB-Rand-Perfect.
- **SAC.** This uses just the policy learned from the RL algorithm, Soft-Actor Critic (SAC), to accomplish the task.
- **Residual.** This method is based off recent residual-learning techniques that combine model-based and reinforcement learning methods [8], [9].

B. Results

The results comparing the different methods is shown in Table I, this table presents the success rate for insertion as well as the average number of steps needed for completion (a step is equivalent to 50 milliseconds of following the same robot command, as our policy is running at 20 Hz), and the percentage that the end-effector ends up in the $\mathcal{S}_u$ and $\hat{\mathcal{S}}_u$ regions over 30 trials. We also present training iteration performance (task success and steps to completion) for the different methods in Figure 5.

MB-Perfect is able to insert 100% of the time, as it has perfect knowledge of the state, and can be seen as an oracle. We can see that taking random actions with MB-Rand-Perfect does not degrade excessively the full performance achieved by MB-Perfect. However, when we used DOPE as the perception system, which has around 2.5 to 3.5 cm of noise and error, the performance of MB-DOPE and MB-Rand-DOPE drops drastically. MB-Rand-DOPE performs 26.6% better than MB-DOPE, as the random actions can help offset the perception error.

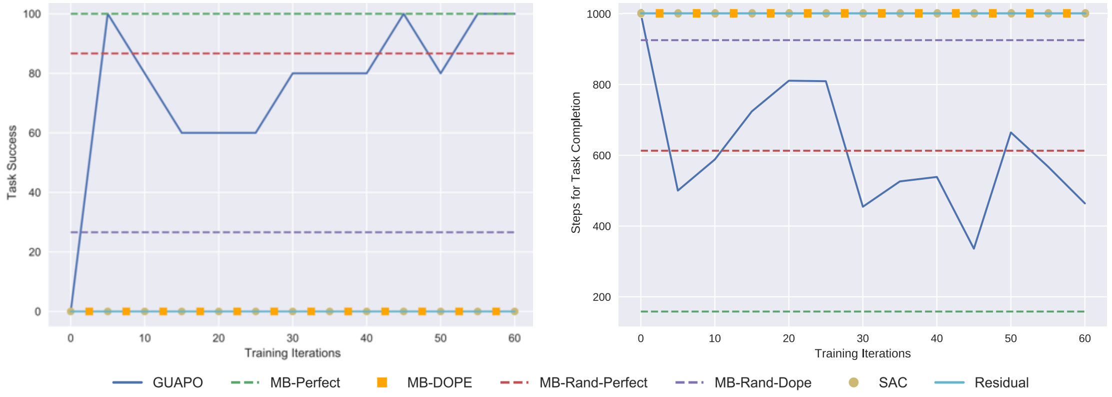
In our setup SAC did not achieve any insertion. This is due to the low number of samples that SAC trained on, since most success stories of RL in the real world require several orders of magnitude more data [27]. The Residual method also did not achieve any insertions. The Residual method often would apply large actions far away from the hole opening, and end up sliding off the box and getting stuck pushing against the side of the box. In comparison, GUAPO only turns on the reinforcement learning policy once it is already nearby the region of interest, and hence does not suffer from this. However, Residual was able to reach $\hat{\mathcal{S}}_u$ 100% of the time after 120 training episodes, while SAC never did.

In comparison, as seen in Fig. 5, after around 8 training iterations, GUAPO is also able to start inserting into the hole (which is about 12 minute real-world training time). As the policy trains, the average number of steps it takes to insert the peg also decreases. After 120 training episodes (and 90 minutes of training), GUAPO is able to achieve 93% insertion rate.

VI. RELATED WORK

In robotic manipulation there are two dominating paradigms to perform a task: leveraging model of the environment (model-based method) or leveraging data to learn (learning-based method). The first category of methods relies on a precise description of the task, such as object CAD models, as well as powerful and sophisticated perception systems [1], [28]. With an accurate model, a well engineered solution can be designed for that particular task [29], [30], or the model can then be combined with some search algorithm like motion planning [31]. This type of model-based approach is limited by the ingenuity of the roboticist, and could lead to irrecoverable failure if the perception system has un-modeled noise and error.

On the other hand, learning-based approaches in manipulation [5], [32] do not require such detailed description, but rather require access to interaction with the environment, as well as a reward that indicates success. Such binary rewards are easy to describe, but unfortunately they render Reinforcement Learning methods extremely sample-inefficient. Hence many prior works use shaped rewards [33], which requires considerable tuning. Other works use low-dimensional state spaces [34] instead of image inputs, which requires either precise perception systems or specially-designed hardware with sensors. There are some proposed methods that manage to deal directly with the sparse rewards, like automatic curriculum generation [15], [16] or the use of demonstrations [35], [36], [37], but these approaches still require large amounts of interaction with the environment. Furthermore, if the position of the objects in the scene changes or there are new distractors in the scene, these methods need to be



(a) Real Robot Task Success

(b) Steps for Peg Insertion

Fig. 5: GUAPO is compared with five other methods: (1) Model-based policy with perfect goal estimate (MB-Perfect), (2) Model-based policy with additive random actions and perfect goal estimate (MB-Rand-Perfect), (3) Model-Based policy with additive random actions using DOPE goal estimates (MB-Rand-DOPE), (4) Reinforcement learning algorithm Soft Actor Critic (SAC), and (5) Residual policy. We run the policy over 60 training iterations, each with two episodes, 1000 steps long.

TABLE I: Real world peg insertion results out of 30 trials. All learning policies (SAC, Residual, and Guapo) trained for 120 episodes (which takes around 90 minutes). The first row indicates percentage success for a full peg insertion. The second row depicts the speed of insertion for the trained policies. The last two rows indicate the percentage the method enters $\mathcal{S}_u$ and $\hat{\mathcal{S}}_u$.

	MB-Perfect	MB-DOPE	MB-Rand-Perfect	MB-Rand-DOPE	SAC [24]	RESIDUAL [9]	GUAPO (ours)
Success Rate	100%	0%	86.67%	26.6%	0%	0%	93%
Avg. Steps for Task Completion	158.3	n/a	554.1	925.4	n/a	n/a	469.6
In $\mathcal{S}_u$	100%	0%	100%	70.0%	0%	0%	100%
In $\hat{\mathcal{S}}_u$	100%	100%	100%	93.3%	0%	100%	100%

fully retrained. On the other hand, our method is extremely sample-efficient with a sparse success reward, and is robust to these variations thanks to the model-based component.

Recent works can also be understood as combining model-based and learning-based approaches. One such method [38] uses a reinforcement learning algorithm to find the best parameters that describe the behavior of the agent based on a model-based template. The learning is very efficient, but at the cost of an extremely engineered pre-solution that also relies on an accurate perception system. Another line of work that allows to combine model-based and learning-based methods is Residual Learning [9], [8], where RL is used to learn an additive policy that can potentially fully over-write the original model-based policy and does not require any further structure. Nevertheless, these methods are hard to tune, and hardly preserve any of the benefits of the underlying model-based method once trained.

The problem of known object pose estimation is a vibrant subject within the robotics and computer vision communities [19], [39], [40], [41], [42], [43], [44], [45], [46]. Regressing to keypoints on the object or on a cuboid encompassing the object seems to have become the defacto approach for the problem. Keypoints are first detected by a neural network, then PnP [47] is used to predict the pose of the object. Peng *et al.* [44] also explored the problem of using uncertainty by leveraging a ransac voting algorithm to find regions where a keypoint could be detected. This approach differs from ours as they do not directly regress to a keypoint probability map, they regress to a vector voting map, where line intersection is then used to find keypoints. Moreover their method does not carry pose uncertainty in the final prediction.

VII. CONCLUSIONS

We introduce a novel algorithm, Guided Uncertainty Aware Policy Optimization (GUAPO), that combines the generalization capabilities of model-based methods and the adaptability of learning-based methods. It allows to loosely define the task to perform, by solely providing a coarse model of the objects, and a rough description of the area where some operation needs to be performed. The model-based system leverage this high-level information and accessible state estimation systems to create a funnel around the area of interest. We use the uncertainty estimate provided by the perception system to automatically switch between the model-based policy, and a learning-based policy that can learn from an easy-to-define sparse reward, overcoming the model and estimation errors of the model-based part. We show learning in the real world of a peg insertion task.

ACKNOWLEDGMENT

Carlos Florensa and Michelle Lee are grateful to all the robotics team at NVIDIA for providing a great learning environment, and providing constant support. Special thanks to Ankur Handa for helping with the compute infrastructure.

References

[1] T. Schmidt, R. Newcombe, and D. Fox, "DART: Dense Articulated Real-Time Tracking," Tech. Rep. [Online]. Available: https://www.cc.gatech.edu/∼afb/classes/CS7495-Fall2014/readings/dart.pdf

[2] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "RMPflow : A Computational Graph for Automatic Motion Policy Generation," Tech. Rep. [Online]. Available: https://arxiv.org/pdf/1811.07049.pdf

[3] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[4] A. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual Reinforcement Learning with Imagined Goals," *Adavances in Neural Information Processing Systems*, 2018.

[5] S. Levine and C. Finn, "End-to-End Training of Deep Visuomotor Policies," vol. 17, pp. 1–40, 2016.

[6] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," *arXiv preprint arXiv:1810.05687*, 2018.

[7] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian Motion Policies," 1 2018. [Online]. Available: http://arxiv.org/abs/1801.02854

[8] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual Policy Learning," 2018.

[9] T. Johannink, S. Bahl, A. Nair, J. Luo, and A. Kumar, "Residual Reinforcement Learning for Robot Control," pp. 1–9.

[10] S. Ganguly and O. Khatib, "Experimental studies of contact space model for multi-surface collisions in articulated rigid-body systems," in *International Symposium on Experimental Robotics*. Springer, 2018.

[11] N. Ratliff, M. Toussaint, and S. Schaal, "Understanding the geometry of workspace obstacles in motion optimization," 2015.

[12] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," 06 2013.

[13] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[14] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal Value Function Approximators," *Internation Conference in Machine Learning*, 2015. [Online]. Available: http://jmlr.org/proceedings/papers/v37/schaul15.pdf

[15] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic Goal Generation for Reinforcement Learning Agents," *International Conference in Machine Learning*, 2018. [Online]. Available: http://arxiv.org/abs/1705.06366

[16] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, "Reverse Curriculum Generation for Reinforcement Learning," *Conference on Robot Learning*, pp. 1–16, 2017. [Online]. Available: http://arxiv.org/abs/1707.05300

[17] Y. Duan, X. Chen, J. Schulman, and P. Abbeel, "Benchmarking Deep Reinforcement Learning for Continuous Control," *International Conference in Machine Learning*, 2016. [Online]. Available: http://arxiv.org/abs/1604.06778

[18] C. Florensa, Y. Duan, and P. Abbeel, "Stochastic Neural Networks for Hierarchical Reinforcement Learning," *International Conference in Learning Representations*, pp. 1–17, 2017. [Online]. Available: http://arxiv.org/abs/1704.03012

[19] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," *arXiv preprint arXiv:1809.10790*, 2018.

[20] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems," 5 2017. [Online]. Available: http://arxiv.org/abs/1705.01292

[21] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, and S. Birchfield, "NDDS: NVIDIA deep learning dataset synthesizer," 2018, https://github.com/NVIDIA/Dataset_Synthesizer.

[22] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, 02 2009.

[23] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks," *arXiv preprint arXiv:1906.08880*, 2019.

[24] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, and C. Sciences, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *Internation Conference in Machine Learning*, pp. 1–15, 2018.

[25] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner, and G. Deepmind, "β-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK," *International Conference in Learning Representations*, pp. 1–22, 11. [Online]. Available: https://openreview.net/forum?id=Sy2fzU9gl

[26] T. Lesort, N. Daz-Rodrguez, J.-F. Goudou, and D. Filliat., "State representation learning for control: An overview," *CoRR*, vol. abs/1802.04181, 2018.

[27] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[28] M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal, "Probabilistic object tracking using a range camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Nov. 2013, pp. 3195–3202.

[29] K. Van Wyk, M. Culleton, J. Falco, and K. Kelly, "Comparative Peg-in-Hole Testing of a Force-Based Manipulation Controlled Robotic Hand," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 542–549, 2018.

[30] C. H. Kim and J. Seo, "Shallow-Depth Insertion: Peg in Shallow Hole Through Robotic In-Hand Manipulation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 383–390, 4 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8598749/

[31] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, P. Abbeel, and R. O. Mar, "Learning Robotic Assembly from CAD," *International Conference on Robotics and Automation*, 2018.

[32] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine, "Composable Deep Reinforcement Learning for Robotic Manipulation," in *Proceedings - IEEE International Conference on Robotics and Automation*, no. 1, 2018, pp. 6244–6251.

[33] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making Sense of Vision and Touch: Self-Supervised Learning of Multimodal Representations for Contact-Rich Tasks," 2018.

[34] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3651–3657.

[35] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards," pp. 1–11, 2017.

[36] Y. Ding, C. Florensa, M. Phielipp, and P. Abbeel, "Goal-conditioned Imitation Learning," *Workshop on Self-Supervised Learning at ICML*, 2019. [Online]. Available: http://arxiv.org/abs/1906.05838

[37] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming Exploration in Reinforcement Learning with Demonstrations," *International Conference on Robotics and Automation*, 2018. [Online]. Available: http://arxiv.org/abs/1709.10089

[38] L. Johannsmeier, M. Gerchow, and S. Haddadin, "A Framework for Robot Manipulation: Skill Formalism, Meta Learning and Adaptive Control," Tech. Rep. [Online]. Available: https://arxiv.org/pdf/1805.08576.pdf

[39] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," in *ACCV*, 2012.

[40] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," in *WACV*, 2017.

[41] S. Zakharov, I. Shugurov, and S. Ilic, "DPOD: Dense 6D pose object detector in RGB images," *arXiv preprint arXiv:1902.11020*, 2019.

[42] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," in *RSS*, 2018.

[43] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, "Segmentation-driven 6D object pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3385–3394.

[44] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *CVPR*, 2019.

[45] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, “Implicit 3d orientation learning for 6d object detection from rgb images,” in *ECCV*, 2018.

[46] B. Tekin, S. N. Sinha, and P. Fua, “Real-time seamless single shot 6D object pose prediction,” in *CVPR*, 2018.

[47] V. Lepetit, F. Moreno-Noguer, and P. Fua, “EP*n*P: An accurate O(*n*) solution to the P*n*P problem,” *International Journal Computer Vision*, vol. 81, no. 2, 2009.