Inverse reinforcement learning for dexterous hand manipulation

Jedrzej Orbik[1], Alejandro Agostini[1,2] and Dongheui Lee[1,3]
[1] Department of Electrical and Computer Engineering
Technical University of Munich, 80333 Munich, Germany
[2] Department of Computer Science
University of Innsbruck, 6020 Innsbruck, Austria
[3] Institute of Robotics and Mechatronics
German Aerospace Center (DLR), 82234 Wessling, Germany

***Abstract*—The success of deep reinforcement learning approaches to learn dexterous manipulation skills strongly hinges on the rewards assigned to actions during task execution. The usual approach is to handcraft the reward function but due to the high complexity of dexterous manipulations the reward definition demands large engineering effort for each particular task. To avoid this burden, we use an inverse reinforcement learning (IRL) approach to automatically learn the reward function using samples obtained from demonstrations of desired behaviours. We have identified that the learned rewards using existing IRL approaches are strongly biased towards demonstrated actions due to the scarcity of samples in the vast state-action space of dexterous manipulation applications. This significantly hinders performance due to unreliable reward estimations in regions unexplored during demonstration. We use statistical tools for random sample generation and reward normalization to reduce this bias. We show that this approach improves learning stability and transferability of IRL for dexterous manipulation tasks. Project page: https://sites.google.com/view/irl-for-dexterous-hand**

***Index Terms*—inverse reinforcement learning, learning from demonstrations, robot skills prototyping, dexterous robotic hand**

I. Introduction

Many robotic applications for manipulation tasks comprise simple end-effectors with a single degree of freedom [1]. This is sufficient for many repetitive industrial tasks such as painting, welding, or assembling, but it limits its applicability in human scenarios where more dexterous manipulation skills are required. Dexterous robotic hands are specially suitable for such scenarios since they permit blending robots into human-centric environments created with ergonomics in mind. Dexterous manipulation does not come without a cost. Because of the high number of degrees of freedom, the control is a demanding task that limits its robotics use. State-of-the-art (SoA) deep reinforcement learning (DRL) algorithms hold promise of autonomous learning with minimal human interaction in high-dimensional domains. However, they require many training samples which may not be feasible to collect only from the hardware. Such data can be also obtained from physically realistic simulators to learn control policies that are lately transferred to real scenarios with additional engineering effort [2]. Because of the complexity of the dexterous hands, the specification of an adequate reward function is another challenge faced by these approaches. Rewards should be carefully crafted for each particular task to achieve a reasonable learning performance but at the expense of limited transferability to new tasks. To overcome this limitation, we use an inverse reinforcement learning (IRL) approach to automatically learn the reward function from demonstrated behaviours for each particular task. We have identified that the sample efficiency of current IRL approaches is not sufficient to deal with the scarcity of demonstrated samples in the extensive state-action space of dexterous manipulation applications. As a consequence, the learned rewards are strongly biased towards demonstrated actions, which deteriorates the overall performance of the system. We borrow tools from statistics and computer vision to mitigate this bias with a three-fold contribution. First, we introduce a random sample generation that improves the robustness of the reward function, exploring regions different from the visited ones during demonstration. Second, we propose a reward normalization that limits the variance of the reward, producing faster and more stable convergence. Third, we propose a reward function dimensionality reduction based on feature selection that improves the sample efficiency. Comparing to the state-of-the-art IRL approach by Fu et al. [3] these improvements are vital for prototyping of human skills on the real-world robots.

II. Related works

Autonomous learning of dexterous manipulation has been a subject to a variety of work. Deep reinforcement learning (DRL) approaches based on policy gradients have received a special attention thanks to their capability of dealing with high-dimensional problems using well-established algorithms. Popov et al. [5] have proposed an extension to Deep Deterministic Policy Gradient algorithm to learn the dexterous manipulation from multiple simple dexterous hands performing the task in parallel and, thus, improving the learning speed. Rajeswaran et al. [4] proposed the Demo Augmented Policy Gradients (DAPG), which is a model-free deep reinforcement learning approach that uses human demonstrations to cope with the exploration difficulties in high dimensional state-action spaces. To assess the performance, they use the MuJuCo simulator [6], which provide physically realistic simulations of dexterous hand in a variety of configurable tasks. Open AI [2] has also used the MuJuCo simulator for learning dexterous object manipulation with Proximal Policy Optimization (PPO) algorithm. This approach has been extended in a later work

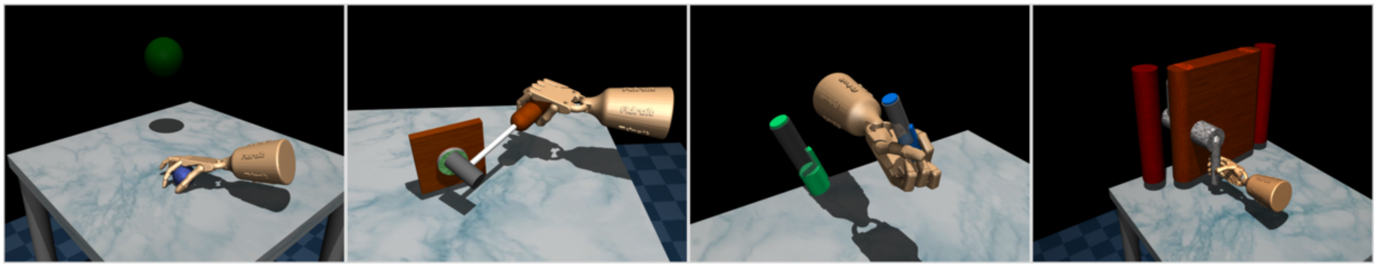
Fig. 1. Dexterous hand manipulation evaluation tasks: object relocation (blue sphere has to be relocated to target green position), tool usage (the hammer is used to place the nail in the board), in-hand manipulation (the pen (blue) is to be rotated to the target orientation presented with the green pen) and door opening [4].

[7], where the automatic domain randomization has been introduced. Other works have focused on learning approaches that provide an alternative to the computationally intensive policy gradient algorithms. Kumar et al. [8] train a set of policies in data-driven manner using the nearest neighbor method for policy selection during evaluation. The work by Falco et al. [9], on the other hand, combines reactive control on the tactile sensor and RL with vision input to learn dexterous hand manipulation. Gupta et al. [10] propose a method that exploits the benefit of human demonstrations to guide the search of optimal policy, avoiding the need of an exhaustive exploration of possibly task-irrelevant regions of the state-action space.

The work on DRL application for dexterous hand manipulation [4] allows the efficient learning, but the real world application is limited by the necessity of the reward function engineering. This may lead to unwanted actor behavior [11] and the manipulation of the environment may even result in highly undesirable reward tampering [12]. This limitation can be tackled by automatically learning the reward function with an inverse reinforcement learning (IRL) approach. A generative adversarial networks (GAN) setting [13] has been proposed by Finn et al. [14] in trajectory-centric approach. An extension to their work was adversarial inverse reinforcement learning (AIRL) by Fu et al. [3], with specific form of discriminator, which disentangles the reward function from the task dynamics. Ho et al. [15] have proposed the generative adversarial imitation learning (GAIL), which directly uses binary classifier as the discriminator for the imitation learning algorithm. This form does not have explicit reward function representation, so it does not give any insight into the task, but the approach can be used to learn the target policy from the demonstrations.

We show that the high variability of the reward values during learning combined with the high dimensionality of the state-action space hinder performance and transferability of current IRL approaches. The learned reward values are only reliable in regions of the state space close to the visited ones during demonstration. In regions far away from samples, the performance greatly deteriorates due to the bias of the reward towards demonstrated actions. This is expected since learning dexterous manipulation requires several high-complexity samples to properly cover the vast state-action space, a requirement difficult to fulfill using only samples obtained from demonstrations. To attack this problem, we borrow tools from computer vision and statistics. The statistical bias is reduced by adding random samples drawn from the probabilistic policy. This strategy is complemented with a reward normalization that diminishes its variance, increasing learning stability. Finally, task-irrelevant variables are masked to reduce the size of the state space. In spite of their simplicity, these tools have never been used to tackle the low sample efficiency of IRL methods for dexterous hand manipulation. As demonstrated by the results, our approach not only permits a significant improvement in the performance but also a better transferability across different challenging tasks.

In the next sections we present the theoretical background of IRL (Sec. III) and of our contributions to IRL for dexterous hand manipulation (Sec. IV). The experimental validation of these contributions is done in Sec. V. The paper ends with some discussions (Sec. VI) and conclusions (Sec. VII).

III. Inverse reinforcement learning

Inverse reinforcement learning, called also Inverse Optimal Control or Apprenticeship Learning, holds the promise of inferring the reward function $r(s_t, a_t)$ from the expert demonstrations given the samples of the expected optimal or near optimal policy π_E (expert policy). The problem to solve is defined as a finite Markov Decision Process (MDP), which is specified by a set $\mathcal{S}$ of Markov states s and a set $\mathcal{A}$ of actions a in time step t, the reward function $r(s_t, a_t)$, transition probability $p(s_{t+1}|s_t, a_t)$ and discount factor $\gamma \in (0,1)$ [16]. The objective of reinforcement learning is the optimization of the policy $\pi_{\boldsymbol{\theta}} \sim p(a_t|s_t)$ defined by the parameters $\boldsymbol{\theta}$ to find the optimal one $\pi^*_{\boldsymbol{\theta}}$, which maximizes the value function $V_{\pi_{\boldsymbol{\theta}}}(s_t)$ for all the states:

$$\pi^*_{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} V_{\pi_{\boldsymbol{\theta}}}(s_t) \quad \forall s_t \in \mathcal{S}, \tag{1}$$

$$V_{\pi_{\boldsymbol{\theta}}}(s_t) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[r(s_t) + \sum_{n=t}^{T} \gamma^{n-t} r(s_n, a_n)\right]. \tag{2}$$

State-action value function is denoted as $Q(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{n=t}^{T} \gamma^{n-t} r(s_n, a_n)\right]$ and advantage function is $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$.

State-of-the-art IRL is implemented in the form of Generative Adversarial Networks (GAN) [13], which consists of two models: generator G and discriminator D competing against each other. The discriminator's aim is to distinguish between the samples from the target distribution (expert demonstrations samples) and the samples from generator (samples generated by the learned policy). The generator is trying to maximize the probability of discriminator to make a mistake, which would take place when the discriminator cannot distinguish between samples generated by the policy and the expert. This two-player contest seeks the saddle point of the expression:

$$\min_D \max_G \mathbb{E}_{x\sim\pi_E}[\log(D(x))] + \mathbb{E}_{x\sim\pi_{\boldsymbol{\theta}}}[\log(1-D(x)]. \quad (3)$$

Finn et al. [14] have proposed a special form of the discriminator:

$$D_{\boldsymbol{\psi}}(\tau) = \frac{\exp\left(f_{\boldsymbol{\psi}}(\tau)\right)}{\exp(f_{\boldsymbol{\psi}}(\tau)) + \pi_{\boldsymbol{\theta}}(a|s)}, \quad (4)$$

in the trajectory τ centric formulation to learn the reward function in inverse reinforcement learning setup. The later work on IRL by Fu et al. [3] proposes a state- and action-centric formulation $f_{\boldsymbol{\psi}}(s_t, a_t)$ that is better suited for practical deep reinforcement learning application. However, despite this latter improvements, all the aforementioned IRL contributions still suffer from a low sample efficiency and lack of robustness of the reward function in high-dimensional domains. That is particularly visible in the complex dexterous manipulation tasks. In this work, we extend the IRL framework by Fu et al. [3] to address these problems. We propose the normalization of the reward function $r(s_t, a_t)$, the addition of random samples in the learning process, as samples from generator's distribution $\pi_{\boldsymbol{\theta}}$ to allow better generalization, and a dimensionality reduction of the input space based on relevant feature selections.

IV. IRL FOR DEXTEROUS HAND MANIPULATION

This section presents the theoretical background of our contributions to tackle the limitations of current IRL applications in dexterous manipulation application. As a baseline DRL algorithm, we use the augmented policy gradient introduced by Rajeswaran et al. [4].

A. Reward normalization

Classically in reinforcement learning, the rewards are assumed to be stationary. This is especially important in the actor-critic setup, where the variance of the gradient is minimized with fitted advantage function $A(s_t, a_t)$ used as a critic [17]. In IRL, however, the non-stationarity of the reward function influences the accuracy of the advantage function estimates and increases the gradient variance. Therefore, in order to limit the magnitude of the advantage function and to control the gradient variance during training, we apply a reward normalization. This normalization stabilizes the learning of the reward function by reducing the variability of the mean and standard deviation of the reward. The importance of normalization of target functions has been studied in the past for Temporal Difference methods [18]. It has lead to simplification of the hyperparameters search and has improved the learning stability. We follow the work by van Hasselt et al. [18] and apply their proposed value target normalization to normalize the rewards according to the formula:

$$r_{\text{norm}}(s,a) = \frac{r(s,a) - \mu_t}{\sigma_t}, \quad (5)$$

where

$$\mu_t = \frac{1}{N}\sum_i^N r(s,a)^{(i)}, \qquad \sigma_t^2 = \frac{1}{N}\sum_i^N \left(r(s,a)^{(i)} - \mu_t\right)^2, \quad (6)$$

at iteration step t for N observations. These formulas can be adapted to the incremental case by the introduction of the step size $\beta \in [0,1]$:

$$\mu_t = (1-\beta)\mu_{t-1} + \beta r(s,a), \qquad \nu_t = (1-\beta)\nu_{t-1} + \beta r(s,a)^2, \quad (7)$$

with ν_t being the second moment. The estimated standard deviation equals $\sigma_t^2 = \nu_t - \mu_t^2$. This yields an exponential moving average that puts more weight on the recent data points as β step size is constant. The initial values of μ_0 and σ_0 are arbitrarily set to 0 and 1 respectively.

B. Feature subspace selection

The generative adversarial networks introduced before seek to solve the problem by mining the generated examples of actor states. But, because of large input space, it is still vulnerable to exploitation if the query from previously unseen input space is provided after training. The lack of robustness of the neural network models due to large input space has been described in previous works, e.g. in computer vision. There, classification models were evaluated as vulnerable both to the synthesized adversarial examples [19] and to the physical world examples after seemingly negligible input variations [20], [21]. The vulnerability of the learned policies in multi-agent environments in reinforcement learning has been described by Gleave et al. [22]. They show how the masking of the observation to the agent-specific features mitigates this effect. After our initial experiments, we have recognized the relevance of the masking also in IRL. The problem of lack of coverage of the input space of the reward function approximator can be seen as an example of adversarial attack on the artificial neural network representing the policy. Here, we follow the view of adversarial examples in work by Goodfellow et al. [23] as "inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake". In our case, the mistake provides high rewards in state and action pairs $r(s,a)$, which does not comply with the expected behavior. This may easily happen if a policy produces the observations in regions that were not explored during reward function training. We propose a simple solution to this problem by masking the input space of the reward function model to the dimensions relevant for the task. As introduced in the Sec. III we can see the IRL in the GAN setup as the two player contest. By application of the masking, similarly as in work in the multi-agent direct RL setup described by Gleave et al. [22], we impede the manipulation of the observations of the other player by the agent and hinder the exploitation of its deficiencies. The agent, in turn, has to keep the access to the whole observation because it is not conditioned on the previous experience in the episode it would not be able to retrieve its state. This enables the agent to retain the optimal policy in the IRL setup.

Using a subspace of task-specific features as the input for the discriminator is enough to produce a meaningful learning signal, resulting in a more robust reward function obtained with fewer training samples. Reduction of spurious state features also reduces amount of the idiosyncrasies of the task, which could be used by discriminator to easily distinguish between the generated samples and the demonstrations that has inferior effect on the training signal. We evaluate different choices of the reward function input masking and its influence on the training performance in Sec. V-C.

C. Generation of Random Samples

We have recognized that the problem of lack of robustness of the neural network model leads to the reward function biased towards demonstrated actions observed during training. To avoid this bias, we propose to provide additional adversarial samples in inverse reinforcement learning as coming from the generator distribution $\pi_{\boldsymbol{\theta}}$ during learning of the reward function. To this end, we have evaluated the Fast Gradient Sign Method (FGSM) method [19], [24] to produce the adversarial samples. However, the random noise samples from the normal distribution estimated on the previously seen samples has performed better. We hypothesize, that this way we produce more plausible samples, which are helpful in the bias reduction. The reward fitting is then implemented as optimization of the following function:

$$\boldsymbol{\psi} = \arg\max_{\boldsymbol{\psi}} \mathbb{E}_{x\sim\pi_E}[\log(D(x))] + \mathbb{E}_{x\sim\pi_{\boldsymbol{\theta}}}[\log(1-D(x)]+ \\ \omega\,\mathbb{E}_{x\sim\mathcal{N}(\mu,\sigma)}[\log(1-D(x)], \tag{8}$$

where ω is the ratio of the random samples from the normal distribution $\mathcal{N}(\mu,\sigma)$ in all generated samples. The parameters of random noise sample distribution μ (mean) and σ (standard deviation) are fitted based on the expert demonstration samples and on samples observed in the previous policy rollouts. The random noise is introduced only for the generated samples to avoid the bias of the obtained policy. We are currently adding the noise to the samples from the policy based on sample variance. These samples are still plausible and help us to generalize better in our MDP representation.

V. Experiments

To assess the validity of our approach, we use the well-established MuJoCo physics engine [6]. This simulator has been widely used in previous approaches [2], [4] and would help to better contrast our results with reference methods. In the experiments, we provide 25 human demonstrations originating from the work by [4] and 25 demonstrations from learned policy delivered with the same paper. The demonstrations represent the behavior that should be exhibited by the actor in each environment setting. As in the conventional learning from demonstration setup, we provide only the successful demonstrations for the imitation, and we do not require any additional samples from the failed trials.

We evaluate the validity of the proposed improvements with respect to SoA IRL algorithms and the quality of the learning in new tasks without additional hyperparameter tuning with exception for in-hand manipulation task, where 1 epoch has been used for behavior cloning. *Behavior cloning*: epochs: 5, learning rate: 1e-3. *Policy (actor)*: number of FC layers: 2, units per layer: 32, activation function: Tanh, step size: 0.1, gamma: 0.995, GAE, lambda: 0.97, trajectories per update: 200. *Value function (critic)*: epochs: 2, batch size: 64, learning rate: 1e-3. *Discriminator*: number of FC layers: 2, units per layer: 63, activation function ReLU, learning rate: 1e-3, batch size: 256, trajectories per update: 200, max updates of generator per discriminator update: 4, steps till max update no. for generator: 150, minimal loss threshold: 0.01. *Random samples*: percentage in all generated samples: 20%.

To provide a reference performance, we contrast the results with those we obtained with the approaches AIRL [3] and GAIL [15]. Finally, for the experimental evaluation, we use four tasks for dexterous hand manipulation: object grasping and relocation, object in-hand manipulation, tool usage and door opening with a dexterous as introduced in the work by Rajeswaran et al. [4]. Fig. 1 depicts snapshots of these tasks.

A. Performance evaluation

We assess the performance obtained by introduction of random samples (Sec. IV-C) and by using the reward normalization (Sec. IV-A). In both cases, we use the masking of the training samples described in Sec. IV-B. The performance evaluation is done for the task of object grasping and relocation. The transferability of the method to other tasks is assessed separately (see Sec. V-B).

1) Generation of Random Samples: The results obtained by addition of generated random samples are shown in Fig. 2. The performance evaluation shows the superiority of our method over the state-of-the-art IRL and comparable results to forward reinforcement learning with engineered reward functions [4]. The addition of random samples generated by the RL system in the training process mitigates the bias towards demonstrated actions and significantly improves the performance. To shed more light on the results of our IRL approach using random samples, we present in Fig. 3 a learned reward function at the end of IRL process. Our method is able to learn a reward function that better discriminates between positive and negative rewards than the vanilla IRL counterparts. This is reflected, for example, in the more defined high reward around the target (black star). We use the fixed learned reward function obtained at the end of the learning process of the example in

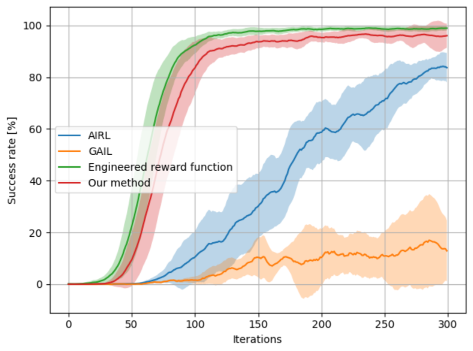

Fig. 2. Comparison of our method to SoA IRL algorithms and forward learning with manually specified reward function, provided for the reference.

Fig. 3 to teach a new policy from scratch using the forward reinforcement learning method DAPG [4]. The results of the evaluation in the tasks are presented in Table I. The results represent the mean of maximum returns from the original reward function calculated over 10 runs. These results show that our method is more robust, exhibits lower overfitting and thus can be used more effectively for learning.

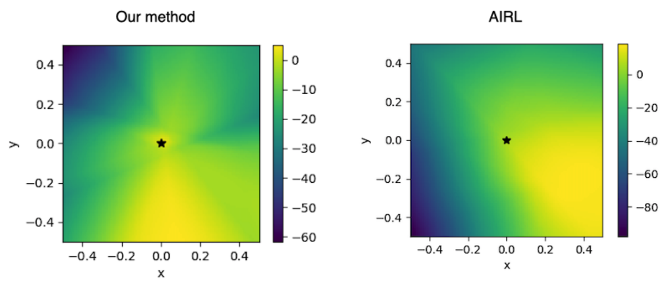

Fig. 3. Learned reward function in object relocation task. The figure presents the sampled values of the learned reward depending on the distance between object and target given in meters. Target position is marked with a star. Our method provides highest reward in the goal position as expected, while the baseline method has high reward regions outside the target area.

2) Reward normalization: The results of the reward normalization are depicted in Table II. They represent mean over multiple runs of the accumulated rewards from the original reward function. The increased stability of learning contributes to higher returns in each of the evaluated dexterous hand manipulation task. Tuning of the reward normalization step size β additionally improves the learning performance. As we can see in the table, the normalization leads to better performance after correct hyperparameter adjustment. The results show that the values of 0.005 or 0.001 are a reasonable choice.

B. Transferability of the method

In order to demonstrate the transferability, the performance of our IRL approach has been evaluated in the four different tasks proposed by Rajeswaran et al. [4]: object relocation, door opening, object in-hand manipulation, and tool usage. The performance evaluation is carried out using the same method from Sec. V-A1, using noise samples and feature masking. In the evaluation, the demonstrations of successful task execution have been provided for our method and the state-of-the-art IRL method [3]. As presented in the provided results in Fig. 4, our method allows the transfer without additional labor while the forward reinforcement learning method would fail in the new tasks without sufficient reward function engineering. Masking of the state space and addition of the noise samples considerably improved the learning performance. With AIRL [3], however, performance deteriorates at the end of the learning process. The reward deficiencies are exploited by the policy by visiting previously non-sampled regions, where overfitted, non-robust reward function is maximized without exhibiting expected behavior. It occurs in the very same way as the human-specified reward can be exploited during learning. Our method provides robust reward function, which prevents such issue from arising throughout the learning process.

C. Ablation study - feature subspace selection

During our experiments, we are using the four dexterous hand manipulation tasks proposed by Rajeswaran et al. [4] together with the state space they defined. For the feature subspace selection we follow a simple rule. We use the features specific to the tasks as the input of the reward function and ignore the dimensions corresponding to the robotic hand. Based on our evaluation, we maintain that the masked reward function still produces the signal necessary for the policy training but significantly reduces sample complexity in most of the tasks. The original, complete state space of object relocation MDP is defined as:

- position and orientation of hand base (6 dimensions)[1]
- hand joint angles (24 dimensions)[2]
- $\overrightarrow{p_O p_H}$ - vector between object and hand (3 dims.) [3]
- $\overrightarrow{p_T p_O}$ - vector between target and object (3 dims.) [3]
- $\overrightarrow{p_T p_H}$ - vector between target and hand (3 dims.) [3]

The dimensions denoted with superscript [3] are task-specific and are provided as the observations to the reward function.In case of the tool usage task, the original state space consists of the following elements:

- 2 DOF position of the hand base (2 dimensions)[1]
- hand joint angles (24 dimensions)[2]
- palm position (3 dimensions)[3]
- nail's head distance from the board (1 dimension)[3]
- nail position (3 dimensions)[3]
- force exert on the nail (1 dimension)[3]
- tool velocity (6 dimension)[3]
- tool position (6 dimensions)[3]

Here, we use the feature subspace describing the palm, tool and nail, since these are particular to the task. Similarly, in the remaining tasks we mask the features, which describe the hand pose and retain only the dimensions specific to the task

[1]hand base features
[2]hand joint angle features
[3]task specific features

	Object relocation	Tool usage	In-hand manipulation	Door opening
AIRL [3]	-2.86	-208.43	32.19	-54.64
Our method w/o random samples	-1.43	-218.42	56.00	-53.36
Our method with random samples	**15.37**	**-110.36**	**63.84**	**-52.15**

TABLE I

RESULTS OF LEARNING FROM SCRATCH ON THE FIXED REWARD FUNCTION AT THE END OF IRL PROCESS.

Norm. step size β	Object relocation	Tool usage	In-hand manipulation	Door opening
0.005	3,103.48	**7,498.09**	59.87	**3,178.89**
0.001	**3,209.82**	4,256.21	**72.24**	3,129.72
0.0005	3,108.84	5,688.66	71.01	2,860.37
0.0001	2,924.39	6,741.54	71.67	2,753.86
W/o normalization	2,597.16	4,485.93	71.60	2,729.51

TABLE II

RESULTS OF THE EVALUATION OF DIFFERENT NORMALIZATION STEP SIZES.

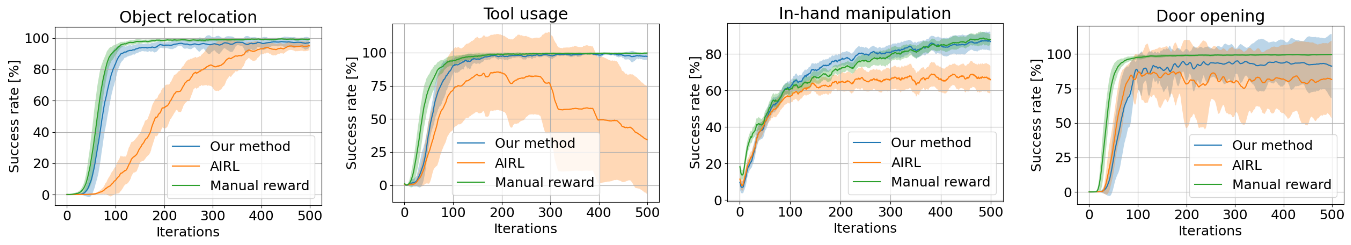

Fig. 4. Performance evaluation for four different tasks. Our IRL method is able to learn efficiently in a new task based only on the provided demonstration. Delivers results comparable with the results of forward learning, where a manual reward function specification is necessary.

indicated with superscript [3]. The in-hand manipulation task includes following features of the state space:

- hand joint angles (24 dimensions)[2]
- object position and orientation (6 dimensions)[3]
- target orientation - randomized (3 dimensions)[3]
- object angular and linear velocities (6 dimensions)[3]
- $\overrightarrow{p_T p_O}$ - vector between target and object, target position is fixed (3 dimensions) [3]
- difference between target θ_T and object angles θ_O (3 dimensions) [3]

The door opening task:

- orientation of the hand base (3 dimensions)[1]
- hand joint angles (24 dimensions)[2]
- latch position (1 dimension)[3]
- door hinge angle (1 dimension)[3]
- palm position (3 dimensions)[3]
- door handle position (3 dimension)[3]
- $\overrightarrow{p_D p_H}$ - vector between door handle and the hand (3 dimensions)[3]
- Indicator variable {-1,1} for closed and open door[3]

The results of the experiments are presented in Fig. 5. The results of this ablation study demonstrate that the reward subspace selection alone is not sufficient to obtain a policy with high success rate and other stability improvement methods are important for the stable convergence. That is particularly visible in the tool usage MDP where the task-specific selection alone is not sufficient and the additional proposed methods are necessary for to the stable convergence of the policy. In this task all the features seem to be relevant to the task and provide results better than using just their subspace. In turn in the task of object relocation we are achieving significantly better results with the task-specific features.The door opening is the least challenging of all the tasks and we can see that the training leads to comparable performance of both task-specific subspace and complete input space.

VI. Discussion

We were able to create the method for learning challenging dexterous hand manipulation directly from the demonstration. For the improvement the key addition was the masking of the state space to task-relevant dimensions. This prevents the inclusion of spurious information, which would deepen the overfitting and facilitate exploitation of the learned model [22]. One could argue that we trade the necessity of the reward function engineering for the feature engineering in state space masking. However, the selection of such features for the complex tasks used in the experiments according to the rule described in Sec. IV-B required a significantly lower effort than the manual engineering of the rewards for such tasks. Finally, the robustness of the reward function has been increased comparing to the vanilla IRL method (as depicted in Fig. 3). However, the obtained robustness is not enough to learn a policy from scratch without the reward function updates due to the lack of the systematized generated samples mining. In the next section we propose some alternative future research lines to tackle this limitation.

VII. Conclusions

We evaluated our method with respect to state-of-the-art inverse reinforcement learning (IRL) methods [3], [15]

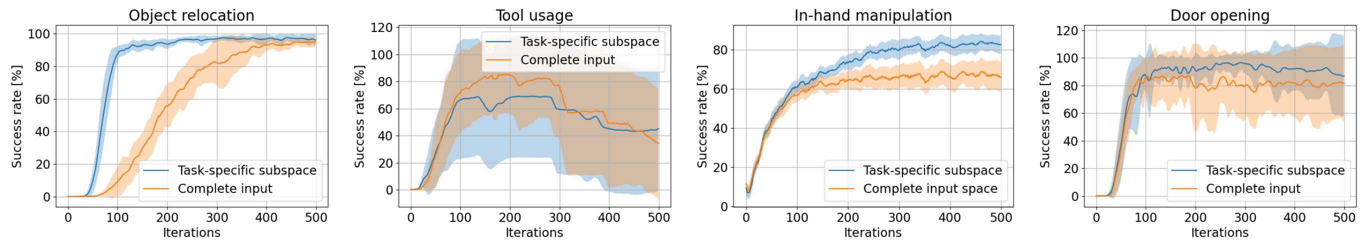

Fig. 5. Subspace selection evaluation for four dexterous manipulation tasks. It is visible that the selection of task specific features improves the performance with respect to using the entire feature space.

and forward reinforcement learning method with manually specified reward function [4] in the tasks of dexterous hand manipulation. Our approach reduces the bias in the reward values towards demonstrated actions in large state-action spaces produced by current IRL approaches by complementing the samples generated from demonstration with samples generated randomly from the learned probabilistic policy. This approach, combined with a reward normalization to reduce the reward variance and with a masking strategy to reduce the dimension of the input space, significantly improves the learning stability and the transferability of existing inverse and forward reinforcement learning approaches. Future research will explore alternative DRL methods that use more informative generation of samples [19], [25], [26] compared to the random sampling method used in this work. Besides this, we will assess the validity of our IRL approach in other robotic domains using simulated and real robot platforms.

ACKNOWLEDGMENT

This work has been partially supported by Helmholtz Association and by the Austrian Science Fund (FWF) Project M2659-N38.

REFERENCES

[1] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, 1st ed. CRC Press, Dec. 2017.

[2] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1808.00177

[3] J. Fu, K. Luo, and S. Levine, "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning," *arXiv:1710.11248 [cs]*, Oct. 2017.

[4] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," *arXiv:1709.10087 [cs]*, Sep. 2017.

[5] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, "Data-efficient Deep Reinforcement Learning for Dexterous Manipulation," p. 12.

[6] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vilamoura-Algarve, Portugal: IEEE, Oct. 2012, pp. 5026–5033.

[7] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[8] V. Kumar, A. Gupta, E. Todorov, and S. Levine, "Learning Dexterous Manipulation Policies from Experience and Imitation," *arXiv:1611.05095 [cs]*, Nov. 2016.

[9] P. Falco, A. Attawia, M. Saveriano, and D. Lee, "On Policy Learning Robust to Irreversible Events: An Application to Robotic In-Hand Manipulation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1482–1489, Jul. 2018.

[10] A. Gupta, C. Eppner, S. Levine, and P. Abbeel, "Learning Dexterous Manipulation for a Soft Robotic Hand from Human Demonstration," *arXiv:1603.06348 [cs]*, Mar. 2017.

[11] J. Clark and D. Amodei, "Faulty Reward Functions in the Wild," https://openai.com/blog/faulty-reward-functions/, Dec. 2016.

[12] T. Everitt and M. Hutter, "Reward Tampering Problems and Solutions in Reinforcement Learning: A Causal Influence Diagram Perspective," *arXiv:1908.04734 [cs]*, Aug. 2019.

[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[14] C. Finn, P. Christiano, P. Abbeel, and S. Levine, "A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models," *arXiv:1611.03852 [cs]*, Nov. 2016.

[15] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," *arXiv:1606.03476 [cs]*, Jun. 2016.

[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, second edition ed., ser. Adaptive Computation and Machine Learning Series. Cambridge, Massachusetts: The MIT Press, 2018.

[17] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[18] H. P. van Hasselt, A. Guez, A. Guez, M. Hessel, V. Mnih, and D. Silver, "Learning values across many orders of magnitude," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 4287–4295.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *arXiv:1412.6572 [cs, stat]*, Mar. 2015.

[20] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[21] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song, "Robust physical-world attacks on machine learning models," *arXiv preprint:1707.08945*, vol. 2, no. 3, p. 4, 2017.

[22] A. Gleave, M. D. C. Wild, N. Kant, and S. L. S. Russell, "Adversarial Policies: Attacking Deep Reinforcement Learning," p. 11, 2019.

[23] I. J. Goodfellow, N. Papernot, S. Huang, R. Duan, P. Abbeel, and J. Clark, "Attacking machine learning with adversarial examples," Feb 2017. [Online]. Available: https://openai.com/blog/adversarial-example-research/

[24] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," *arXiv preprint arXiv:2001.03994*, 2020.

[25] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," *arXiv:1706.06083 [cs, stat]*, Sep. 2019.

[26] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial Training for Free!" *arXiv:1904.12843 [cs, stat]*, Nov. 2019.