Leveraging Physical Models for Gentle Manipulation

Sandy H. Huang
University of California, Berkeley
shhuang@cs.berkeley.edu

Martina Zambelli, Yuval Tassa, Jackie Kay,
Murilo F. Martins, Patrick M. Pilarski, Raia Hadsell
DeepMind
{zambellim, tassa, kayj, murilomartins, ppilarski, raia}@google.com

Abstract

Robots must know how to be *gentle* when they need to interact with fragile objects, or when the robot itself is prone to wear-and-tear. We propose an approach that enables deep reinforcement learning to train policies that are gentle. This involves augmenting the (task) reward with a penalty for non-gentleness. However, augmenting with only this penalty impairs learning: policies get stuck in a local optimum of avoiding all contact with the environment. Introducing surprise-based intrinsic rewards solves this problem, as long as the right kind of surprise is chosen—penalty-based surprise is more effective than the typical dynamics-based surprise. Videos are available at http://sites.google.com/view/gentlemanipulation.

1 Introduction

Deploying deep RL on real-world robots often leads to substantial wear-and-tear over time, on both the robot itself and the environment. If robots were able to minimize excessive forces and impacts while learning, they would last longer before needing repairs, and the objects they interact with would not need to be replaced as often.

Thus, in order to broadly deploy deep RL on real robots, we need an approach for training policies that are gentle. A naive approach is to constrain the torques that a robot's motors can exert. However, many manipulation tasks require occasional or momentary high force (e.g., hammering a nail); the torque limit cannot be any lower than this. But we do not want the robot to freely exert this much force along its entire trajectory. Alternatively, one could constrain the total amount of force allowed, but this requires knowing *a priori* the minimum total amount necessary to complete the task [2, 4].

Instead, our approach is to give the robot negative rewards for actions that are not gentle, for instance those that result in high-impact forces. This is a natural way to encode preferences about *how* robots should perform a task (e.g., driving style [1]), and can be seen as an intrinsic "pain" signal that encourages learning safer policies. However, perhaps unsurprisingly, we show adding only this penalty makes learning much harder: agents get stuck in a local optimum of avoiding contact altogether, because they encounter the penalties before task reward, and thus learn a fear of pain.

To motivate agents to interact with the environment *and* do it gently, we propose balancing this "pain" signal by adding another intrinsic signal, this one positive, for curiosity. In particular, we reward the agent for *surprising* experiences—those that contradict its current physical model of the world. A concrete example is rewarding transitions that have low probability under a learned dynamics model [3, 7]. However, we find using this kind of *dynamics-based surprise* is not as effective as using a *penalty-based surprise*, that leads robots to be explicitly curious about the non-gentleness penalty itself. Using the latter enables precise task execution and successful manipulation of fragile objects.

2 Proposed Approach

We define being gentle as minimizing impact. To train policies for gentle manipulation, we augment the original reward (r) from the task-specific Markov Decision Process with an impact force penalty (r^f) and a surprise-based intrinsic reward (r^s). Agents are trained to maximize $r' = r + r^f + r^s$.

2.1 Impact penalty

We define impact m_t to be $\max(0, f_{t+1} - f_t)$, where f_t is the sensed force at time step t. In other words, for a robot to be gentle, it should minimize *increases in sensed force*.[1] The impact force penalty acts as an intrinsic pain signal. It scales non-linearly with the level of impact, by taking into account the *acceptability*, specified by $a_\lambda(m) \in [0, 1]$, of a particular amount of impact m. This is a monotonically increasing function, that should be designed according to how resilient the robot and environment are to impact. In our experiments, we set $a_\lambda(m) = \text{sigmoid}(\lambda_1(-m + \lambda_2))$.

The impact penalty at time step t is $r_t^f = -\sum_i a_\lambda(m_t^i) * m_t^i$; the summation is across force sensors at different locations on the robot (e.g., the fingers of a robot hand). In our experiments, $\lambda = [2, 2]^\top$.

2.2 Surprise-based intrinsic reward

If the environment reward consists of only the task reward and impact penalty, that is, $r'_t = r_t + r_t^f$, we find that policies get reliably stuck in a local optimum of not making contact with anything in the environment—the agent learns to be afraid of contact, since it encounters the impact penalty before the sparse task reward, hindering exploration. The purpose of adding surprise-based intrinsic rewards is to encourage policies to make *contact* with objects in the environment but still in a *gentle* way.

Dynamics-based. For an agent to be "surprised," it must have some (learned) understanding of its environment, i.e., a model. In the case of dynamics-based surprise, this model is a dynamics model that takes in the current state and action, and predicts the next state. We train an ensemble of neural networks for the dynamics model, in order to have predictive uncertainty [8] to capture novelty.

Each of the M networks in the ensemble outputs the mean and variance of a Gaussian for each dimension d of the prediction. The ensemble's combined output is a mixture of Gaussians for each output dimension d. During training, each network is randomly initialized, and they are trained on different batches of transitions. We choose $M = 5$, as recommended by related research [8].

To compute dynamics-based surprise intrinsic reward r_t^s, we approximate the dynamics model's predicted distribution over next states with a single Gaussian per output dimension d, to measure how much variance there is *across* networks in the ensemble. The intrinsic reward is the negative log-likelihood of the true next state under this predicted distribution over next states.

Penalty-based. Perhaps counter-intuitively, we propose to reward the agent for being curious about the impact penalty itself, by adding a reward to focus the learning and exploration of the agent on the intrinsic pain signal, thus enabling better prediction of pain through gentle interaction. To compute this penalty-based surprise reward $r_t^{s_p}$, we train an impact penalty predictor in parallel with the agent, with the same implementation as the general dynamics model (an ensemble of five neural networks).

We compute the intrinsic reward differently though—the problem with directly using negative log-likelihood is that then areas of high penalty are acceptable, as long as the prediction likelihood in those areas is low. However, this leads to excessive non-gentle behavior. To enforce the preference for exploring areas with low penalty while avoiding ones with high penalty, a natural approach is to augment the task reward with a convex combination between the negative log-likelihood and the impact penalty. In this convex combination, the weight on the negative log-likelihood is $a_{\lambda'}(r_t^f) \in [0, 1]$, which specifies the acceptability of a particular penalty r_t^f. This acceptability is a monotonically increasing function, that should be chosen based on how much penalty the robot may experience for the sake of exploration or task completion. We use a sigmoid function for this, as we did for impact $a_\lambda(m)$ (Sec. 2.1). In addition, we only provide this intrinsic reward if the penalty is non-zero, because the purpose is to encourage the agent to (cautiously) learn more about the penalty.

[1] As motivation, consider a robot that needs to apply a force of 20N to push a heavy object. If the robot increases the amount of force it applies from zero to 20N in a fraction of a second, the impact results in more potential wear-and-tear, compared to increasing gradually to 20N over several seconds.

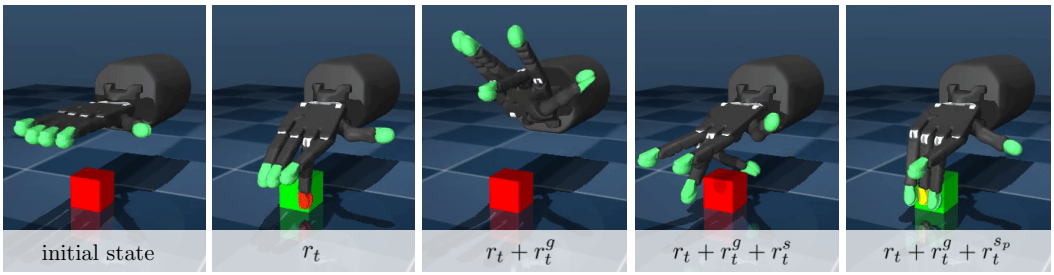

Figure 1: The task is to apply at least 5N of force to the block; green block indicates success. Fingertip color shows the amount of impact force, from yellow (near-zero) to red (10N). Policies were trained for 500k iterations.

2.3 Implementation details

We train policies with Distributed Distributional Deterministic Policy Gradients (D4PG) [6], an actor-critic algorithm for continuous control. We use a batch size of 256 and a replay buffer of 1 million transitions. The actor network has two fully-connected (fc) layers of size 300 and 200, with a `tanh` after the latter. Its output is delta position; this is added to the current position to obtain the action. We train a separate critic network per reward type, for more stable learning [10]. Each has two fc layers of size 400 and 300; the distributional output has support $(-100, 100)$ and 101 bins. The first hidden layer is followed by layer normalization [5] and `tanh`, and all others by ELU activations.

For the models, each ensemble consists of five neural networks, with three fc layers of 128 hidden units each, followed by ReLU activations. Intrinsic rewards are computed with respect to a target model, which is updated every 5000 (dynamics-based) or 1000 (penalty-based) iterations; this makes training more stable, since the agent is no longer trying to surprise a model that is constantly changing. We start providing intrinsic rewards after 20,000 training iterations, once the model is more accurate.

3 Experiments

Our goal is to learn policies that are safer, with less forceful impacts, while also improving sample efficiency and overall task performance. The following experiments compare three possible approaches: augmenting the task reward with an impact penalty (r^f), this penalty and a dynamics-based surprise reward ($r^f + r^s$), or this penalty and a penalty-based surprise reward ($r^f + r^{s_p}$).

Domain. Our experiments are on a simulated Shadow Dexterous Hand [11] in MuJoCo [12]. It has five fingers with a total of 24 degrees of freedom, actuated by 20 motors. Each fingertip has a spatial touch sensor, with a spatial resolution of 4×4 and three channels: one for normal force and two for tangential. We take the absolute value and then sum across the spatial dimensions, to obtain a 3D force vector for each fingertip. The impact force m_t^i is then the sum over the increase in force per channel for fingertip i. The state consists of proprioception (joint position and joint velocity) and touch. The action space is 20-dimensional. We use position control and a control rate of 20 Hz.

The simulation task involves touching a single block (Fig. 1). Focusing on this simple task enables us to clearly characterize how well the three approaches can train low-impact policies. We find that even in this simple environment, most approaches struggle to learn policies for gentle manipulation.

Manipulation with impact penalty. The task is to press the block with any fingerpad, with a force of at least 5N, at which point the episode terminates with a reward of +1. This task is simple: without an impact penalty, agents learn this task quickly (Fig. 2, left), although with a significant amount of impact (Fig. 1, top center). However, once impact penalties are added, if there is no form of surprise-based intrinsic rewards to counteract them, then agents fail to learn the task—they get trapped in a local optimum of avoiding contact with the environment, since they experience penalties from contact before discovering how to perform the task (Fig. 1).

When impact penalties are present, dynamics-based surprise leads to the policy exploring interesting hand configurations, but with limited touching. In contrast, penalty-based surprise is much more effective in terms of agents learning how to perform the task gently, with low impacts (Fig. 2, right). Even more, these agents learn as quickly as ones trained without the impact penalty (Fig. 2, left).

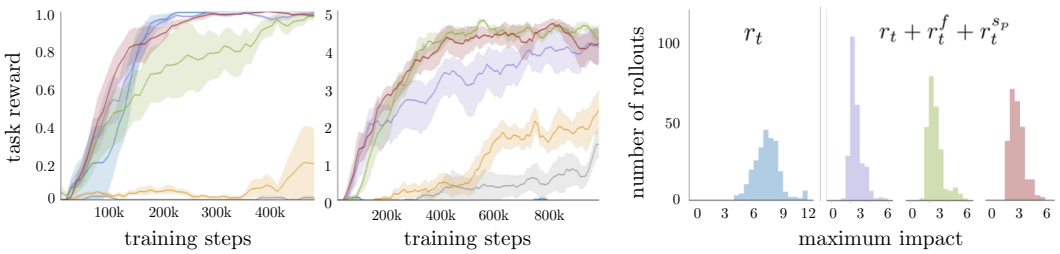

Figure 2: Learning curves for different reward augmentations (five random seeds each) and two tasks: pressing a block (**left**) or a *fragile* block (**center**) with at least 5N of force. Histograms (**right**) show the maximum impact (in Newtons) experienced per rollout, when the agent performs the task successfully. Rollouts are collected after 500k training steps. Policies are trained on: task reward only ■; task reward with impact penalty and no intrinsic rewards ■, dynamics-based surprise ■, or penalty-based surprise. The parameterization λ' for acceptability of penalties varies: $\lambda'_2 = 1.5$ ■, 2 ■, or 3 ■ (left, right) and $\lambda'_2 = 1$ ■, 1.5 ■, or 2 ■ (center). $\lambda'_1 = 2$ for all.

Manipulation of fragile objects. We then made the task more difficult by introducing a fragile block, that breaks if the impact force at any point is greater than 3N, and the episode terminates with a negative reward of -0.5. The reward for completing the task is +5. Now, policies trained with only the task reward are unable to learn the task at all, because they accidentally break the block a few times, and learn that any contact with the block is undesirable. In contrast, policies trained with the impact penalty are better able to learn the task. As before, penalty-based surprise intrinsic rewards are more effective than dynamics-based ones in terms of how quickly policies are able to learn the task (Fig. 2, center).

4 Discussion and Future Work

Our work takes a step toward using deep RL to train policies for gentle, contact-rich manipulation. We found that choosing the appropriate focus of curiosity is important for incentivizing agents to interact gently with the environment, in the presence of impact penalties. This enables precise task execution and successful manipulation of fragile objects.

A main direction of future work is to apply this approach to more complex tasks, in particular tasks in real-world and/or dynamic environments. It would also be interesting to consider model-based approaches that use the learned dynamics and non-gentleness prediction models. In addition, this work only considers one aspect of being gentle—impact. Our approach could be used to train policies while minimizing other sources of wear and tear, for instance total force (rather than the increase in force), or the torques exerted by a robot's motors (which would reduce energy consumption too [9]).

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

[2] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In *ICML*, 2017.

[3] J. Achiam and S. Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.

[4] E. Altman. *Constrained Markov Decision Processes*. CRC Press, 1999.

[5] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[6] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. TB, A. Muldal, N. Heess, and T. P. Lillicrap. Distributed distributional deterministic policy gradients. In *ICLR*, 2018.

[7] R. Houthooft, X. Chen, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Vime: Variational information maximizing exploration. In *NIPS*, 2016.

[8] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, 2017.

[9] A. Mohammed, B. Schmidt, L. Wang, and L. Gao. Minimizing energy consumption for robot arm movement. *Procedia CIRP*, 25:400 – 405, 2014.

[10] S. Russell and A. L. Zimdars. Q-decomposition for reinforcement learning agents. In *ICML*, 2003.

[11] Shadow. Shadow Dexterous Hand. `https://www.shadowrobot.com/products/dexterous-hand/`.

[12] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *IROS*, 2012.