





“Good Robot!”: Efficient Reinforcement Learning for Multi-Step Visual Tasks with Sim to Real Transfer

Andrew Hundt , Benjamin Killeen , Nicholas Greene, Hongtao Wu , Heeyeon Kwon, Chris Paxton , and Gregory D. Hager 

Abstract—Current Reinforcement Learning (RL) algorithms struggle with long-horizon tasks where time can be wasted exploring dead ends and task progress may be easily reversed. We develop the SPOT framework, which explores within action safety zones, learns about unsafe regions without exploring them, and prioritizes experiences that reverse earlier progress to learn with remarkable efficiency. The SPOT framework successfully completes simulated trials of a variety of tasks, improving a baseline trial success rate from 13% to 100% when stacking 4 cubes, from 13% to 99% when creating rows of 4 cubes, and from 84% to 95% when clearing toys arranged in adversarial patterns. Efficiency with respect to actions per trial typically improves by 30% or more, while training takes just 1-20 k actions, depending on the task. Furthermore, we demonstrate direct sim to real transfer. We are able to create real stacks in 100% of trials with 61% efficiency and real rows in 100% of trials with 59% efficiency by directly loading the simulation-trained model on the real robot with no additional real-world fine-tuning. To our knowledge, this is the first instance of reinforcement learning with successful sim to real transfer applied to long term multi-step tasks such as block-stacking and row-making with consideration of *progress reversal*. Code is available at https://github.com/jhu-lcsr/good_robot.

Index Terms—Computer vision for other robotic applications, deep learning in grasping and manipulation, reinforcement learning.

I. INTRODUCTION

MULTI-STEP robotic tasks in real-world settings are notoriously challenging to learn. They intertwine learning the immediate physical consequences of actions with the need to understand how these consequences affect progress towards the overall goal. Furthermore, in contrast to traditional motion planning, which assumes perfect information and known action models, learning only has access to the spatially and temporally limited information from sensing the environment.

Manuscript received February 24, 2020; accepted July 20, 2020. Date of publication August 11, 2020; date of current version August 27, 2020. This letter was recommended for publication by Associate Editor J. Kober and Editor T. Asfour upon evaluation of the reviewers' comments. This work was supported by the NSF NRI Awards nos. 1637949 and 1763705, and in part by Office of Naval Research Award N00014-17-1-2124. (Corresponding author: Andrew Hundt.)

Andrew Hundt, Benjamin Killeen, Nicholas Greene, Hongtao Wu, Heeyeon Kwon, and Gregory D. Hager are with The Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: ahundt@jhu.edu; killeen@jhu.edu; ngreen29@jhu.edu; hww67@jhu.edu; hkwon28@jhu.edu; hager@cs.jhu.edu).

Chris Paxton is with NVIDIA, Seattle, WA, 98105 USA (e-mail: cpaxton@nvidia.com).

This article has supplementary downloadable material available at <https://ieeexplore.ieee.org>, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3015448

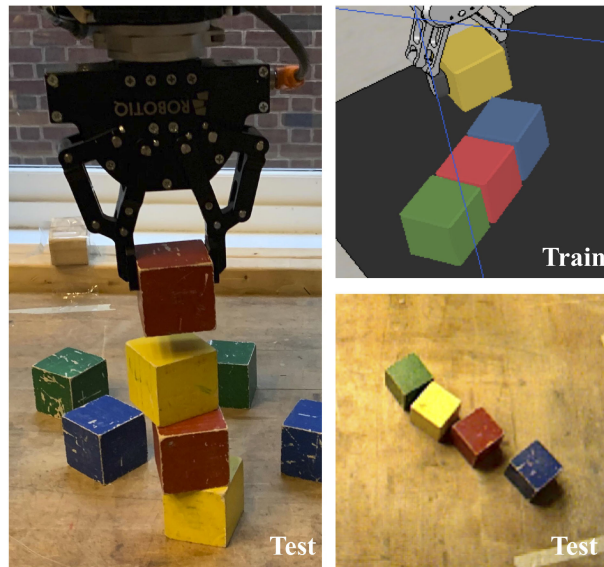


Fig. 1. Robot-created stacks and rows of cubes with sim to real transfer. Our Schedule for Positive Task (SPOT) framework allows us to efficiently find policies which can complete multi-step tasks. Video overview: <https://youtu.be/MbCuEZadkIw>

Our key observation is that reinforcement learning wastes significant time exploring actions which are unproductive at best. For example, in a block stacking task (Fig. 1), the knowledge that grasping at empty air will never snag an object is “common sense” for humans, but may take some time for a vanilla algorithm to discover. To address this, we propose the Schedule for Positive Task (SPOT) framework, which incorporates common sense constraints in a way that significantly accelerates both learning and final task efficiency.

While these types of constraints are intuitive, incorporating them into Deep RL (DRL) in a manner that leads to reliable and efficient learning is nontrivial [1], [2]. Our methods (Section III) take inspiration from a humane and effective approach to training pets sometimes called “Positive Conditioning.” Consider the goal of training a dog “Spot” to ignore an object or event she finds particularly interesting on command. Spot is rewarded with treats whenever partial compliance with the desired end behavior is shown, and simply removed from regressive situations with zero treats (reward). One way to achieve this is to start with multiple treats in hand, place one treat in view of Spot, and, if she eagerly jumps at the treat (a negative action), the human snatches and hides the treat immediately for zero reward on that action. With repetition, Spot will eventually hesitate, and so she is immediately praised with “Good Spot!” and gets a treat

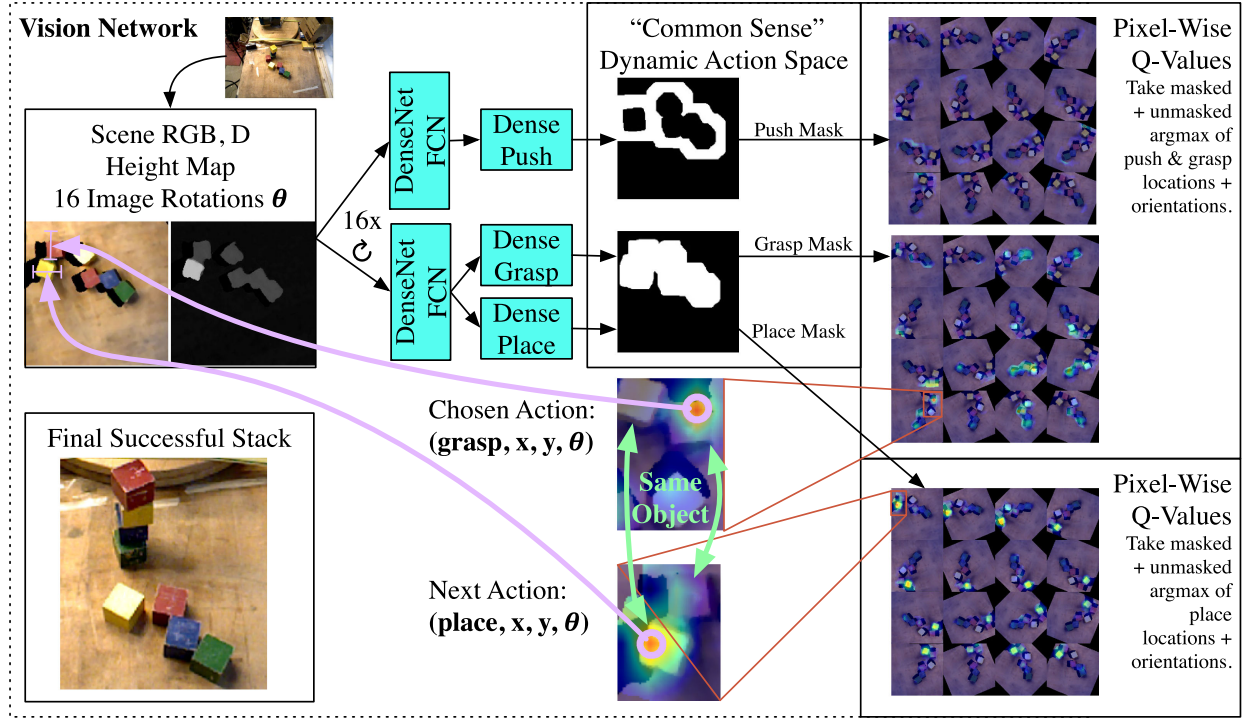


Fig. 2. Our model architecture. Images are pre-rotated to 16 orientations θ before being passed to the network. Every coordinate $a = (\phi, x, y, \theta)$ in the output pixel-wise Q-Values corresponds to a final gripper position, orientation, and open loop action type, respectively. Purple circles highlight the highest likelihood action $\arg \max_a (Q(s, M(a)))$ (8) with an arrow to the corresponding height map coordinate, showing how these values are transformed to a gripper pose. The rotated overhead views overlay the Q value at each pixel from dark blue values near 0 to red for high probabilities. If you take a moment to compare the Q values of a single object across all actions (green arrows identify the same object across two oriented views) you will see each object is scored in a way which leads to a successful stack in accordance with its surrounding context. For example, the grasp model learns to give a high score to the lone unstacked red block for grasp actions and a low score to the yellow top of the stack, while the place model does the reverse. Here the model chooses to grasp the red block and place on the yellow, blue, and green stack. Experiment details are in Sections IV and V.

separate from the one she should ignore. This approach can be expanded to new situations and behaviors, and it encourages exploration and rapid improvement once an initial partial success is achieved. As we describe in Section III, our reward functions and SPOT-Q Learning are likewise designed to provide neither reward nor punishment for actions that reverse progress.

Instances of *progress reversal* are associated with varying complexity. On the one hand, failing to stack the first block on top of another leaves the robot in a similar situation, so recovery takes $\Omega(1)$ actions. However, once a stack of n blocks exists, even a successful grasp might knock the whole stack down, reversing the entire history of actions for a given trial (Fig. 3), so recovery is $\Omega(n)$. The latter, more dramatic instance of *progress reversal* is a challenging problem for reinforcement learning of multi-step tasks in robotics; our work provides a method for efficiently solving such cases.

In summary, our contributions in this article are:

- 1) The overall SPOT framework for reinforcement learning of multi-step tasks, which improves on state of the art in simulation and can train efficiently on real-world situations.
- 2) SPOT-Q Learning, a method for safe and efficient training in which a mask focuses exploration at runtime and generates extra on-the-fly training examples from past experience during replay.
- 3) State of the art zero-shot domain transfer from simulated stacking and row building tasks to their real world counterparts, as well as robustness with respect to a change in hardware and scene positions.

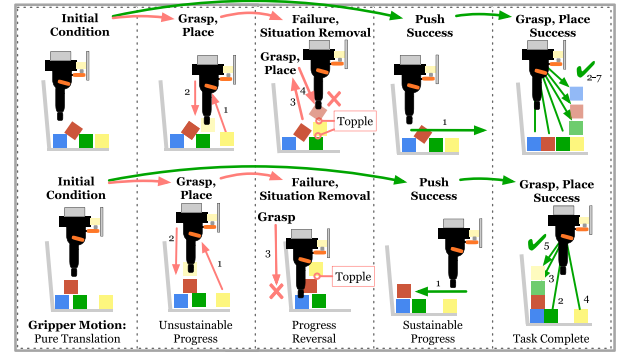


Fig. 3. Red arrows show how individual successful actions can fail on the larger stacking task, forcing eventual *progress reversal* where a partial stack topples or the top must be removed. Ideally algorithms should efficiently learn to prevent this situation and succeed as indicated by the green arrows. Thus, temporal and workspace dependencies must be considered. Events at a current time $t_i \in T$, $i \in [1..n]$ can influence the likelihood of successful outcomes for past actions $t_h | h < i$ and future actions $t_j | j > i$. A successful choice of action at any given t_i will ensure both past and future actions are productive contributors to the larger task at hand. In our experiments a partial stack or row is itself a scene obstacle. The gray wall pictured here is for illustrative purposes only.

- 4) An ablation study showing that Situation Removal dramatically decreases *progress reversal*; that a progress metric increases efficiency; and that trial rewards improve on discounting, but involve a trade-off between efficiency and support for sparse rewards.

II. RELATED WORK

Deep Neural Networks (DNNs) have enabled the use of raw sensor data in robotic manipulation [1]–[5]. In some approaches, a DNN’s output directly corresponds to motor commands, e.g., [3], [4]. Higher-level methods, on the other hand, assume a simple model for robotic control and focus on bounding box or pose detection for downstream grasp planning [1], [6]–[11]. RGB-D sensors can be beneficial [1], [11], [12], as they capture physical information about the workspace. Object-centric skill learning can be effective and generalize well, e.g., [13]–[16] focus on block stacking by classifying simulated stacks as stable or likely to fall. Similarly, [17], [18] develop physical intuition by predicting push action outcomes. Our work differs by developing visual understanding and physical intuition in concert with the progress of multi-step tasks.

Grasping is a particularly active area of research. DexNet [19], [20] learns from a large number of depth images of top-down grasps, and gets extremely good performance on grasping novel objects but does not look at long-horizon tasks. 6-DOF GraspNet [21] uses simulated grasp data to generalize to new objects and has been extended to handle reliable grasping of novel objects in clutter [12].

DRL has proven effective at increasingly complex tasks in robotic manipulation [1], [5], [22], [23]. QT-Opt [5] learns manipulation skills from hundreds of thousands of real-world grasp attempts on real robots. Domain Adaptation, such as applying random textures in simulation, can also enhance sim to real transfer [24], [25]. Other methods focus on transferring visuomotor skills from simulated to real robots [22], [26]. Our work directs a low-level controller to perform actions rather than regressing torque vectors directly, following prior work [1], [23] by learning a pixel-wise success likelihood map.

Multi-step tasks with sparse rewards present a particular challenge in reinforcement learning because solutions are less likely to be discovered through random exploration. When available, demonstration can be an effective method for guiding exploration [27]–[29]. Multi-step tasks can be split into modular sub-tasks comprising a sketch [30], while [31] has robot-specific and task-specific learning modules.

Safety is crucial for reinforcement learning in many real-world settings [32]–[34]. The preliminary experiments in Section IV-D show that SPOT-Q provides a way to incorporate safety into general Q-Learning based algorithms [35].

We compare the SPOT framework to VPG [1], a method for RL-based table clearing tasks which can be trained from images within hours on a single robot, in Sections IV and V. VPG is frequently able to complete adversarial scenarios like first pushing a tightly packed group of blocks apart and then grasping the now-separated objects.

Some of the most closely related recent work involves tasks with multiple actions: [36] includes placing one block on another, [37] places one towel on a bar, and [38] clears a bin, but the first two are not long-horizon tasks and the possibility of *progress reversal* (Fig. 3) is never considered.

III. APPROACH

We investigate multi-step tasks for which there is a sparse and approximate notion of task progress. It is possible to improve the efficiency of learning by taking these four measures: structuring such problems to capture invariant properties of the data, deploying traditional algorithms where they are most effective,

ensuring rewards do not propagate through failed actions, and introducing an algorithm which removes unnecessary exploration. We will later demonstrate our approach in the context of the general problem of assembly through vision-based robotic manipulation.

We frame the problem as a Markov Decision Process (S, A, P, R) , with state space S , action space A , transition probability function $P : S \times S \times A \rightarrow \mathbb{R}$, and reward function $R : S \times A \rightarrow \mathbb{R}$. This includes a simplifying assumption equating sensor observations and state. At time step t , the agent observes state s_t and chooses an action a_t according to its policy $\pi : S \rightarrow A$. The action results in a new state s_{t+1} with probability $P(s_{t+1}|s_t, a_t)$. As in VPG [1], we use Q-learning to produce a deterministic policy for choosing actions. The function $Q : S \times A \rightarrow \mathbb{R}$ estimates the expected reward R of an action from a given state, i.e. the “quality” of an action. Our policy π selects an action a_t as follows:

$$\pi(s_t) = \arg \max_{a \in A} Q(s_t, a) \quad (1)$$

Thus, the goal of training is to learn a Q that maximizes R over time. This is accomplished by iteratively minimizing $|Q(s_t, a_t) - y_t|$, where the target value y_t is:

$$y_t = R(s_{t+1}, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1})) \quad (2)$$

Q-learning is a fundamental algorithm in RL, but there are key limitations in its most general form for applications like robotics where the space and cost of actions and new trials is extremely large, and efficient exploration can be essential or even safety critical. It is also highly dependent on R , whose definition can cause learning efficiency to vary by orders of magnitude, as we show in Section IV-C, and so we begin with our approach to reward shaping.

A. Reward Shaping

Reward shaping is an effective technique for optimizing a reward R , to train policies [39] and their neural networks efficiently. Here, we present several reward functions for later comparison (Section IV-C), which build towards a general formulation for reward shaping conducive to efficient learning on a broad range of novel tasks, thus reducing the *ad hoc* nature of successful reward schedules.

Suppose each action a is associated with a sub-task $\phi \in \Phi$ and that we have an indicator function $\mathbb{1}_a[s_{t+1}, a_t]$ which equals 1 if an action a_t succeeds at ϕ and 0 otherwise.¹ As in VPG [1], our baseline rewards follow this principle and include a sub-task weighting function $W : \Phi \rightarrow \mathbb{R}$, according to their subjective difficulty and importance:²

$$R_{\text{base}}(s_{t+1}, a_t) = W(\phi_t) \mathbb{1}_a[s_{t+1}, a_t]. \quad (3)$$

Next, we define a sparse and approximate task progress function $\mathcal{P} : S \rightarrow \mathbb{R} \in [0, 1]$, indicating proportional progress towards an overall goal, where $\mathcal{P}(s_t) = 1$ means the task is complete.³ As in our story of Spot the dog (Section I), a *progress reversal* leads us to perform Situation Removal (SR) on the agent and

¹Examples of action indicator sources include the grasp detector in our Robotic 2F85 gripper, human supervision, or another detection algorithm.

²In our experiments we assign simple values for each successful action type: $W_{\phi_t} \in \{W_{\text{push}} = 0.1, W_{\text{grasp}} = 1, W_{\text{place}} = 1\}$.

³In our block tasks \mathcal{P} is the height of the stack or length of the row vs the goal size, in table clearing either the number of objects or occupied pixels vs the total, and in navigation the remaining vs initial distance.

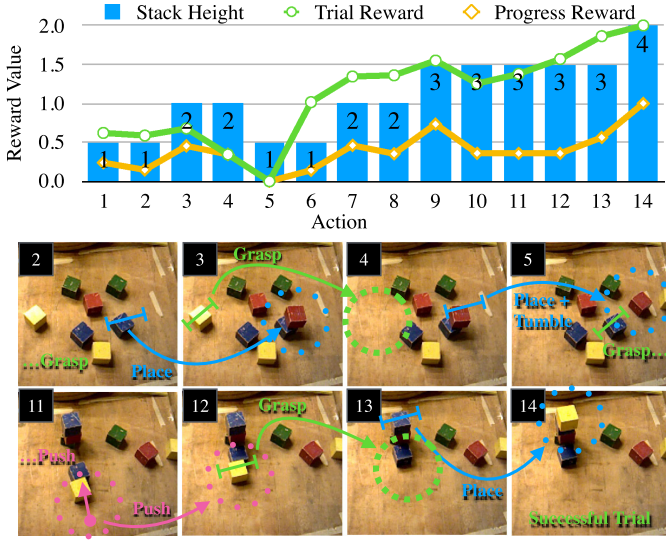


Fig. 4. Example of SPOT Trial Reward $\mathbb{R}_{\text{trial}}$ (6), and the SPOT Progress Reward R_p (5) with images of key action steps. Actions 1–3: a_1 is an initial grasp, followed by a successful place where a slightly off balance stack of height 2 is formed. Actions 4–5: *Progress reversal* occurs when a grasp then place knocks the stack over, so the reward values go to zero. Action 7: While not pictured, the scene is similar to a_3 but with a better balanced top block. Intuitively, since a_9 doesn’t topple like a_5 a better reward at a_7 would be appropriate, which is one advantage of $\mathbb{R}_{\text{trial}}$ over R_p , because $R_p(s_4, a_3) = R_p(s_8, a_7)$ and $\mathbb{R}_{\text{trial}}(s_4, a_3) < \mathbb{R}_{\text{trial}}(s_8, a_7)$ because a_7 leads directly to a successful stack. Actions 11–14: Grasp and place actions lead to a full stack of 4 completing the trial. The final $\mathbb{R}_{\text{trial}}$ at a_{14} is $2 \times R_p$. Here $W_{\phi_t} \in \{W_{\text{push}} = .5, W_{\text{grasp}} = 1, W_{\text{place}} = 1.25\}$ for chart visibility.

physically reset the environment during training (Fig. 3). We define an associated indicator $\mathbb{1}_{\text{SR}}[s_t, s_{t+1}]$, which equals 1 if $\mathcal{P}(s_{t+1}) \geq \mathcal{P}(s_t)$ and 0 otherwise. These lead to new reward functions:

$$R_{\text{SR}}(s_{t+1}, a_t) = \mathbb{1}_{\text{SR}}[s_t, s_{t+1}] R_{\text{base}}(s_{t+1}, a_t) \quad (4)$$

$$R_p(s_{t+1}, a_t) = \mathcal{P}(s_{t+1}) R_{\text{SR}}(s_{t+1}, a_t) \quad (5)$$

One advantage of R_{base} , R_{SR} , and R_p is that each is available “instantaneously” in the midst of a trial after two state transitions. However, they do not consider the possibility that an early mistake might lead to failure many steps down the line (Fig. 3, 4), and so we will develop a reward which propagates across whole trials.

B. Situation Removal: SPOT Trial Reward

Is it possible for a reward function to account for actions which lead to failures at a later time step while still training more efficiently than a standard discounted reward \mathbb{R}_D where $\mathbb{R}_D(s_{t+1}, a_t) = \gamma \mathbb{R}_D(s_{t+2}, a_{t+1})$? Our approach is to block reward propagation through failed actions via the Situation Removal concept:

$$\mathbb{R}_{\text{trial}}(s_{t+1}, a_t) = \begin{cases} 0, & \text{if } R_*(s_{t+1}, a_t) = 0 \\ 2R_*(s_{t+1}, a_t), & \text{if } t = N \\ R_*(s_{t+1}, a_t) + \gamma \mathbb{R}_{\text{trial}}(s_{t+2}, a_{t+1}), & \text{otherwise} \end{cases} \quad (6)$$

Algorithm 1: SPOT-Q with Prioritized Experience Replay.

```

1: Input Replay Memory
    $H_T = (S_T, A_T, R_T, \text{Predicted}_T)$ 
2: while AGENT_IS_ACTING( ) do
3:    $t = \text{PRIORITIZED\_EXPERIENCE\_SAMPLE}(T, H_T)$ 
4:    $y_t = R(s_{t+1}, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1}))$ 
5:    $\delta_t = \text{HUBER\_LOSS}(Q(s_t, a_t); y_t)$ 
6:    $a_{\pi,t} = \pi(s_t)$ 
7:   if  $M(s_t, a_{\pi,t}) = 0$  then  $\triangleright$  The action would fail.
8:      $y'_t = \gamma Q(s_{t+1}, a_{\pi,t})$   $\triangleright$  New 0 reward sample.
9:      $\delta_t = \delta_t + \text{HUBER\_LOSS}(Q(s_t, a_t); y'_t)$ 
10:  end if
11:   $\text{BACKPROP}(\sum \delta_t)$ ; step optimizer; update weights.
12: end while

```

where R_* can be an arbitrary instant reward function such as R_{SR} or R_p from Section III-A, N marks the end of the trial, and γ is the usual discount factor which is set to $\gamma = 0.65$.

The effect of using $\mathbb{R}_{\text{trial}}$ is that future rewards only propagate across time steps where subtasks are completed successfully. As illustrated in Fig. 4 and described in the caption, the zero reward from situation removal cuts the propagation of future rewards back through time steps containing failed actions. This focuses learning on short and successful sequences that complete a task.

C. SPOT-Q Learning and Dynamic Action Spaces

In this section, we go a step further and leverage *a priori* knowledge about the environment to make simple but powerful assumptions which both reduce unproductive attempts and accelerate training. Specifically, there are many occasions when certain action failures are easily predicted from the same sensor signal used for Q learning. To this end, we assume the existence of an oracle, $M(s_t, a) \rightarrow \{0, 1\}$, which takes the *current* state s_t and an action a and returns 0 if an action is certain to fail, and 1 otherwise. This is subtly different from the success indicator $\mathbb{1}_a[s_{t+1}, a_t]$, which requires the *outcome* s_{t+1} of an action a_t to determine success or failure.⁴ Using M , we define the dynamic action space $M_t(A)$:

$$M_t(A) = \{a \in A | M(s_t, a) = 1\}. \quad (7)$$

In short, $M_t(A)$ does not tell us whether $a \in A$ is an action worth taking, but rather whether it is worth exploring.

Given a state s_t , the question becomes how to most effectively utilize M_t in training. If $\pi(s_t) \notin M_t(A)$, then $\pi(s_t)$ can be treated as a failure for the purposes of learning *and* we can explore the next best action not guaranteed to fail. To formalize this, we introduce **SPOT-Q Learning** which is a new target value function replacing (2):

$$y_{M,t} = \begin{cases} y_t, & \text{if } \pi(s_{t+1}) \in M_t(A) \\ y_t + \gamma Q(s_{t+1}, \pi_M(s_{t+1})) + R(s_{t+1}, a_t), & \text{otherwise} \end{cases} \quad (8)$$

⁴For example, grasping an object can only succeed if there is depth data in the neighborhood of a predicted action, so attempts to grasp in free space can be easily predicted to fail, as we demonstrate in Section IV

TABLE I
SIMULATED ROBOT TASK RESULTS (SECTION IV) WITH THE SPOT FRAMEWORK

Simulation Stack of 4	SPOT-Q	Mask	Reward	Trials	Efficiency
Discounted $R_t = \gamma R_{t+1}$	\times	\times	$R_{\mathcal{D}}$	0%	0%
Discounted $R_t = \gamma R_{t+1}$	\times	\checkmark	$R_{\mathcal{D}}$	0%	0%
Baseline R_{base} eq. 3	\times	\times	R_{base}	2-13%	1-2%
Situation Rem. R_{SR} eq. 4	\times	\times	R_{SR}	90-95%	8-23%
Task Prog. $R_{\mathcal{P}}$ eq. 5	\times	\times	$R_{\mathcal{P}}$	98-98%	38-52%
Trial Rew. R_{trial} eq. 6	\times	\times	R_{trial}	95-97%	30-32%
Mask but no SPOT-Q	\times	\checkmark	R_{trial}	95%-99%	46%-55%
SPOT-Q + R_{trial}	\checkmark	\checkmark	R_{trial}	100-100%	45-51%
SPOT-Q + $R_{\mathcal{P}}$	\checkmark	\checkmark	$R_{\mathcal{P}}$	96-100%	25-45%
Simulation Row of 4	SPOT-Q	Mask	Reward	Trials	Efficiency
Baseline R_{base} eq. 3	\times	\times	R_{base}	0-13%	0-1%
Situation Rem. R_{SR} eq. 4	\times	\times	R_{SR}	94-98%	19-43%
Task Prog. $R_{\mathcal{P}}$ eq. 5	\times	\times	$R_{\mathcal{P}}$	96-98%	34-57%
Trial Rew. R_{trial} eq. 6	\times	\times	R_{trial}	74-87%	11-20%
Mask but no SPOT-Q	\times	\checkmark	R_{trial}	92-93%	16-32%
SPOT-Q + R_{trial}	\checkmark	\checkmark	R_{trial}	94-94%	25-34%
SPOT-Q + $R_{\mathcal{P}}$	\checkmark	\checkmark	$R_{\mathcal{P}}$	98-100%	62-68%

Multi-Step Task Test Success Rates Measured Out of 100% for Simulated Tasks Involving Push, Grasp and Place Actions Trained for 20 K Actions (Section IV-C). Bold Entries Highlight Our Key Algorithm Improvements Over the Baseline. “Trials” Indicates the Overall Rate at Which Stacks or Rows Are Successfully Completed. the Algorithm Components Are Described in Section III, Except for “Mask But No SPOT-Q” Which Is a Special Case Described in the SPOT-Q Section of Our Ablation Study (Section IV-C). Values Are the Min and Max of Two Runs.

where $\pi_M(s_t) = \arg \max_{a \in M_t(A)} Q(s_t, a)$. Crucially, we perform backpropagation on both the masked action, which has 0 reward, and the unmasked action $\pi_M(s_t)$, which the robot actually performs.

Algorithm 1 describes how we continuously train from past examples with SPOT-Q and Prioritized Experienced Replay (PER) [40] as the current policy is rolled out. In Section IV, we discuss how SPOT-Q allows us to surpass prior work, wherein similar heuristics [1], [41] neither match SPOT-Q nor account for the safety considerations we discuss later.

IV. SIMULATION EXPERIMENTS

Our method improves performance and action efficiency over the state of the art on the table clearing task from VPG [1], as well as on two challenging multi-step tasks of our design: creating a stack of four blocks and creating a horizontal row of four blocks. Our best results can achieve 100% trial success on the simulated stacking and row tasks, models which successfully transfer to the real world as we show in Section V.

We detail a series of simulation experiments to understand the contribution of each element of our approach to this overall performance. To do so, we evaluate each reward function, the effect of SPOT-Q on heuristic exploration, other possible SPOT-Q implementations, the reward weighting term W , and then we describe our best results with SPOT-Q + $R_{\mathcal{P}}$ and SPOT-Q + R_{trial} . In brief, we find that Situation Removal R_{SR} is the largest contributor to our improved performance, $R_{\mathcal{P}}$ improves accuracy and efficiency, and R_{trial} trains more efficiently than discounted rewards while accounting for a time delay between actions and consequences. SPOT-Q improves results over no masking, and over basic masking on its own. Finally, we test a grid world navigation task [42] to show how the SPOT framework applies to safe reinforcement learning. Tables I and III summarize these results.

TABLE II
REAL ROBOT TASK RESULTS (SECTION V) WITH THE SPOT FRAMEWORK

Real Test Task	Domain Train	Trials Test	Action Complete	Efficiency	Reward	SPOT-Q	Training Actions
Clear 20 Toys	Real	Real	1/1	75%	R_{trial}	\checkmark	1k
Stack of 4	Real	Real	82%	60%	R_{trial}	\checkmark	2.5k
Stack of 4	Sim	Real	100%	51%	$R_{\mathcal{P}}$	\times	20k
Stack of 4	Sim	Real	100%	61%	$R_{\mathcal{P}}$	\checkmark	20k
Stack of 4	Sim	Real	100%	61%	R_{trial}	\checkmark	20k
Row of 4	Sim	Real	90%	83%	$R_{\mathcal{P}}$	\times	20k
Row of 4	Sim	Real	100%	59%	$R_{\mathcal{P}}$	\checkmark	20k
Row of 4	Sim	Real	90%	58%	R_{trial}	\checkmark	20k

Bold Entries Highlight Sim to Real Transfer With SPOT-Q. In This Table No SPOT-Q Also Means No Masking.

TABLE III
SAFETY GRID WORLD (FIG. 5) COMPARISON OF ALGORITHM CHANGES ON TOP OF RAINBOW [35]

Mask	SPOT-Q	$R_{\mathcal{P}}$	Trials Complete	Efficiency	Actions
\times	\times	\times	10.9-12.2%	11-12%	> 500k
\checkmark	\times	\times	93.9-96.9%	62-80%	100-145k
\checkmark	\checkmark	\times	95.0-96.0%	72-74%	100-125k
\checkmark	\checkmark	\checkmark	99.8-99.9%	62-64%	65-75k

Cases without $R_{\mathcal{P}}$ Use the Built-In Reward. “Trials Complete” Is the Percentage of 1000 Test Trials Successfully Completed By Reaching the Green Square in Fewer Than 100 Actions Without Entering Lava. “Efficiency” Is the Best Test Ideal/Actual Actions Per Trial After 500 K Training Actions. The Ideal Action Count for Each Trial Is Found Via a Wavefront Planner. “Actions” Reports How Many Training Steps Were Taken Until the First Case Where 100% of 30 Validation Trials Succeed. Values Are the Min and Max of Two Runs.

A. Robot Implementation Details

We consider a robot capable of being commanded to a specified arm pose and gripper state in its workspace. Our action space consists of three components: action types Φ , locations $X \times Y$, and angles Θ . The agent observes the environment via a fixed RGB-D camera, which we project so that z is aligned with the direction of gravity, as shown in Fig. 2. We discretize the spatial action space into a square height map with 0.448m on a side and 224×224 bins with coordinates (x, y) , so each pixel represents roughly 4mm^2 as per VPG [1]. The angle space $\Theta = \{\frac{2\pi i}{k} | i \in [0, k-1]\}$ is similarly discretized into $k = 16$ bins. The set of action types consists of three high-level motion primitives $\Phi = \{\text{grasp}, \text{push}, \text{place}\}$. In our experiments action success is determined by our gripper’s sensor for grasp, object perturbations for push, and an increase in stack height or row length for place.

A traditional trajectory planner executes each action $a = (\phi, x, y, \theta) \in A$ on the robot. For grasping and placing, each action moves to (x, y) with gripper angle $\theta \in \Theta$ and closes or opens the gripper, respectively. A push starts with the gripper closed at (x, y) and moves horizontally a fixed distance along angle θ . Fig. 2 visualizes our overall algorithm, including the action space and corresponding Q -values.

B. Evaluation Metrics

We evaluate our algorithms in randomized test cases in accordance with the metrics found in VPG [1]. Ideal Action Efficiency is 100% and calculated as $\text{Ideal}/\text{Actual}$ action count; defined

as 1 action per object for grasping tasks; and 2 actions per object for tasks which involve placement. This means 6 total actions for a stack of height 4 since only 3 objects must move, and 4 total actions for rows by placing two blocks between two endpoints. We validate simulated results twice with 100 trials of novel random object positions.

C. Algorithm Ablation

We compare the contribution from each component of the underlying algorithm and against baseline approaches in Table I, except for clearing tasks which are provided in the text. Unless otherwise stated we summarize rows and stacks together as a combined average below.

Clear 20 Toys: We establish a baseline via the primary simulated experiment found in VPG [1], where 20 toys with varied shapes must be grasped to clear the robot workspace. The SPOT framework matches VPG [1] with 100% task completion and improves both the rate of grasp successes from 68% to 84% and action efficiency from 64% to 74%.

Clear Toys Adversarial: The second baseline scenario is the 11 cases of challenging adversarial arrangements from VPG [1], where toys are placed in tightly packed configurations. Each case is run 10 times and the SPOT framework completely clears 7/11 cases compared to 5/11 in VPG [1]; the clearance rate across all 110 runs improves to 95% from 84%. Efficiency in this case drops from 60% to 38%, which is accounted for by the increase in the number of difficult cases solved, as separating the blocks can take several attempts.

Reward Functions: R_{base} , R_{SR} , $R_{\mathcal{P}}$, and R_{trial} incrementally extend one another (Section III-A, III-B). All masking is disabled for this study unless otherwise indicated.

$R_{\mathcal{D}}$ s.t. $R_{\mathcal{D}}(s_{t+1}, a_t) = \gamma R_{\mathcal{D}}(s_{t+2}, a_{t+1})$ is discounting, the most conventional approach to trial rewards. When evaluated with $R_{\mathcal{P}}$ at the final time step and $\gamma = 0.9$, grasp and place actions succeed at a rate of 5% and 45%, respectively. Stacks of height 2-3 are created and performance improves with masking (32%, 48%). However, this approach is incredibly inefficient with no stacks of 4 within 20 k actions. That said, we would expect convergence if orders of magnitude more training were feasible [43].

R_{base} is effective for pushing and grasping [1], but it is not sufficient for multi-step tasks, only completing 13% of rows and stacks with about 200+ actions per trial in the best case. In another case, it repeatedly reverses progress by often looping grasping then placing of the same object at one spot, leading to 99% successful grasps but 0 successful trials overall, even after manual scene resets. We do not expect R_{base} to converge on these tasks as there is no progress signal to indicate, for example, that grasping from the top of an existing stack is a poor choice.

R_{SR} resolves the progress reversal problem immediately since such actions get 0 reward; and thus we see an astounding increase in trial successes from 13% to 94%, and an order of magnitude efficiency increase to 23% across both tasks, or about 22 actions per trial.

$R_{\mathcal{P}}$ leads to a rise in combined trial successes to 97%, and efficiency to 45%, or about 20 actions per trial. This improves upon pure situation removal by incorporating the quantitative amount of progress.

R_{trial} utilizes $R_{\mathcal{P}}$ as the instant reward function in this test, and has an average trial success rate of 96% for stacks and efficiency of 31%, or about 19 actions per trial. However, performance

degrades significantly for rows, declining to an 80% trial success rate and just 16% action efficiency, or about 25 actions per trial. These values indicate R_{trial} strikes a trade-off between the inefficiency of $R_{\mathcal{D}}$ and the need for a more instantaneous progress metric in $R_{\mathcal{P}}$, as the most recent value can be utilized to fill actions with no progress feedback. We also note that once SPOT-Q is added this reward is the best for stacking and second best overall, as we show below.

SPOT-Q: VPG [1] evaluated heuristics that specify exact locations to explore, and they found it led to *worse* performance. A similar approach in QT-Opt [41] is phased out as training proceeds, indicating that their methods do not contribute to improving outcomes throughout the training process. By contrast, SPOT-Q is enabled at all times and excises regions with zero likelihood of success, while other regions of interest remain open for exploration. So does this difference in heuristic design matter?

The “Mask but no SPOT-Q” test disables the *if* statement in Alg. 1 to simulate a typical heuristic in which exploration is directed to particular regions without zero reward guidance. “Mask but no SPOT-Q” completes 95% of trials, compared to 88% without masking and 99% with SPOT-Q; action efficiency results are even more pronounced at 37%, 23%, and 50% respectively. Both these results and Section IV-D show SPOT-Q simply works throughout training and testing with little to no tuning, and so we conclude that SPOT-Q improves the efficiency of learning from heuristic data.

SPOT-Q Alternatives: We evaluated two alternatives to SPOT-Q (eq. 8, Alg. 1), where 0 reward backpropagation is performed on all masked pixels with loss applied to the (1) sum, and (2) average of the masked scores in addition to the actually executed action. In both cases, the gradients exploded and the algorithm did not converge. Only SPOT-Q is able to efficiently enhance convergence.

Reward Weighting: SPOT-Q + $R_{\mathcal{P}}$ where $W_{\text{push}} = 0.1$ succeeds in 99% of trials, but just 27% when $W_{\text{push}} = 1.0$. The weighting in Fig. 4 on R_{trial} without masking or SPOT-Q achieves 97% stack success and 38% action efficiency, but we leave all weighting constant for consistency in Table I. This shows $W(3)$ is important for efficient training.

SPOT-Q + $R_{\mathcal{P}}$: This configuration has the best overall simulation performance with a 99% trial success rate and 50% efficiency, or about 10 actions per trial. It is also the best simulated row model with 98% trial success in one test and 100% in the second, with a high 62–68% action efficiency.

SPOT-Q + R_{trial} : This has the best stacking model with 100% completion in both test cases, and 45–51% efficiency. Overall performance is the second best with 97% trial success, and 37% efficiency, or about 14 actions per trial.

D. Safety and Domain Generalization

To demonstrate the broad scope of the SPOT framework, we evaluate it on the simple but challenging Safety Grid World [42] (Fig. 5), an environment type widely used to evaluate RL algorithms [32], [39]. Here the red robot must move forward or turn as it navigates towards the green square without ever entering the lava. If we had just one real robot to learn within this world, standard DRL would be extremely unsafe, but the SPOT framework allows the robot to safely explore the space.

As Table III shows, all improvements are consistent with our more realistic tasks. We start with Rainbow [35], a Q learning

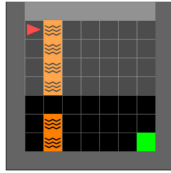


Fig. 5. Safety Grid World where the goal is to avoid lava and get to the green square.

based DRL method, which only completes at most 12% of trials within 500 k actions with a 12% efficiency. We then perform a small ablation study, successively adding Masking, SPOT-Q, and R_P to Rainbow; 96.9%, 95.5%, and 99.9% of 1000 test trials are completed, respectively; average efficiency is 75%, 73%, and 62%, respectively; and the average number of actions to complete 100% of 30 validation trials is 123 k, 113 k, and 70 k, respectively.⁵ All failures with a mask did not enter the lava, they hit a 100 action limit.

These results are consistent with our more realistic experiments, demonstrate how the SPOT framework generalizes across completely different scenarios, and illustrate the application of the SPOT framework to safe exploration. Next, we demonstrate how the SPOT framework leverages knowledge acquired in simulation directly on a real robot task.

V. REAL WORLD EXPERIMENTS

Finally, we examine the performance of SPOT-Q on real robot tasks, both via training from scratch and sim to real transfer. In both cases, performance was roughly equivalent to that achieved in simulation, which shows the strength of our approach for efficient and effective reinforcement learning. We use the setup described in [29], [44], including a Universal Robot UR5, a Robotiq 2-finger gripper, and a Primesense Carmine RGB-D camera; all but the arm differ from those in our simulation. Other implementation details are as described in Section IV-A, and results are in Table II.

Real Pushing and Grasping: We train the baseline pushing and grasping task from scratch in the real world, test with 20 objects and see 100% test clearance, 75% grasp success rate, and 75% efficiency in 1 k actions; these results are comparable to the performance charted by VPG [1] over 2.5 k actions. Sim to real does not succeed in this task.

Sim to Real vs Real Stacking: After training in simulation we directly load the model for execution on the real robot. Remarkably, all tested sim to real stacking models complete 100% of trials, outperforming a model trained on the real robot which is successful in 82% of trials (Fig. 6, Table II). R_P and R_{trial} have an equal action efficiency at 61%, and the version of R_P without SPOT-Q or a mask exhibits slightly lower efficiency at 51%. This is particularly impressive considering that our scene is exposed to variable sunlight. Intuitively, these results are in part due to the depth heightmap input in stacking and row-making.

Sim to Real Rows: Our R_P + SPOT-Q sim to real rows model is also able to create rows in a remarkable 100% of attempts with 59% efficiency. R_{trial} + SPOT-Q and R_P with no mask

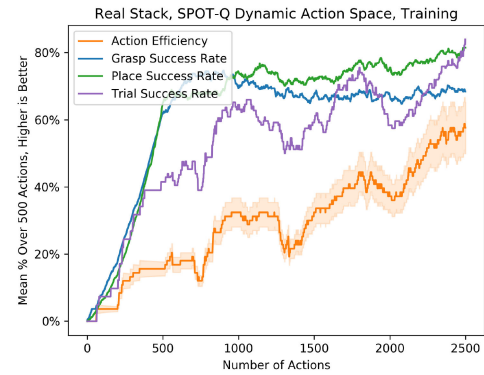


Fig. 6. Real training of the SPOT framework to Stack 4 Cubes with R_{trial} and SPOT-Q. Failures include missed grasps, off-stack placements, and actions in which the stack topples. Toppling can occur during successful grasp and push actions.

perform slightly worse, both with 90% of trials complete, and an efficiency of 83% and 58%, respectively. The high efficiency of R_P with no mask is because we end real trials immediately when the task becomes unrecoverable, such as when a block tumbles out of the workspace. We exclusively evaluate sim to real transfer in this case because training progress is significantly slower than with stacks.

We expect that block based tasks are able to transfer because the network relies primarily on the depth images, which are more consistent between simulated and real data. This might reasonably explain why pushing and grasping does not transfer, a problem which could be mitigated in future work with methods like Domain Adaptation [24], [25].

VI. CONCLUSION

We have demonstrated that the SPOT framework is effective for training long-horizon tasks. To our knowledge, this is the first instance of reinforcement learning with successful sim to real transfer applied to long term multi-step tasks such as block-stacking and creating rows with consideration of *progress reversal*. The SPOT framework quantifies an agent's progress within multi-step tasks while also providing zero-reward guidance, a masked action space, and situation removal. It is able to quickly learn policies that generalize from simulation to the real world. We find these methods are necessary to achieve a 100% completion rate on both the real block stacking task and the row-making task.

SPOT's main limitation is that while intermediate rewards can be sparse, they are still necessary. Future research should look at ways of learning task structures that incorporate situation removal from data. In addition, the action space mask M is currently manually designed; this and the lower-level open loop actions might be learned as well. Another topic for investigation is the difference underlying successful sim to real transfer of stacking and row tasks when compared to pushing and grasping. Finally, in the future, we would like to apply our method to more challenging tasks.

ACKNOWLEDGMENT

We extend our thanks to Adit Murali for Safety Grid World integration; to Molly O'Brien for valuable discussions, feedback,

⁵In the grid world we only evaluate R_P and the built-in reward (where all reward is delivered at the end) because there is little distinction between a failed action and failed trial.

and editing; to Corinne Hundt for the “Good Robot!” title copy-writing; to Michelle Hundt, Thomas Hundt, and Ian Harkins for editing; to all those who gave their time for reading, reviewing, and feedback; and to the VPG [1] authors for releasing their code.

REFERENCES

[1]邢红梅, 陈欣, 王慧. 基于 LightGBM 模型的文本分类研究[J]. 内蒙古工业大学学报: 自然科学版, 2020, 39(1): 52-59.

[2]刘智锋,吴亚平,王继民. 人工智能生成内容技术对知识生产与传播的影响[J].情报杂志,2023,42(07):123-130.

[3]陈永伟. 超越 ChatGPT: 生成式 AI 的机遇, 风险与挑战[J]. 山东大学学报(哲学社会科学版), 2023, 3: 127-143.