Stochastic Grounded Action Transformation for Robot Learning in Simulation (DRAFT)

Siddharth Desai[1], Haresh Karnan[1], Josiah P. Hanna[2], Garrett Warnell[3] and Peter Stone[4]

***Abstract*— Robot control policies learned in simulation do not often transfer well to the real world. Many existing solutions to this *sim-to-real* problem, such as the Grounded Action Transformation (GAT) algorithm, seek to correct for—or *ground*—these differences by matching the simulator to the real world. However, the efficacy of these approaches is limited if they do not explicitly account for stochasticity in the target environment. In this work, we analyze the problems associated with grounding a deterministic simulator in a stochastic real world environment, and we present examples where GAT fails to transfer a good policy due to stochastic transitions in the target domain. In response, we introduce the *Stochastic Grounded Action Transformation* (SGAT) algorithm, which models this stochasticity when grounding the simulator. We find experimentally—for both simulated and physical target domains—that SGAT can find policies that are robust to stochasticity in the target domain.**

I. INTRODUCTION

Learning robot control policies in simulation [1] is typically safer, cheaper, and faster than learning in the real world, but it also introduces a *reality gap* [2] between the training environment (the simulator) and the deployment environment (the real world). *Sim-to-real* algorithms, which focus on overcoming this gap, have recently received a great deal of attention.

This paper focuses on a class of solutions referred to as grounding algorithms [3] which use a small amount of real world data to improve (or *ground*) the simulator. We can assume that the ungrounded simulator at least approximates the correct real world dynamics, so, many grounding approaches [3], [4], [5], [6] learn parameters to correct for these differences. Since it may not always be feasible to modify the internal parameters of a simulator, we focus our attention on grounding algorithms that treat the simulator as a black-box, such as the Grounded Action Transformation (GAT) algorithm [6].

In most robotics domains, the dynamics are often best modeled as stochastic processes due to effects like friction, gear backlash, uneven terrain, and other sources of noise in the environment; however many earlier approaches for sim-to-real do not explicitly account for stochasticity in the real world. These approaches learn a single value for the correction terms when learning a distribution would more accurately reflect the real world.

[1] The University of Texas at Austin, Department of Mechanical Engineering {sidrdesai,haresh.miriyala}@utexas.edu

[2]The University of Edinburgh josiah.hanna@ed.ac.uk

[3]Army Research Laboratory garrett.a.warnell.civ@mail.mil

[4]The University of Texas at Austin, Department of Computer Science pstone@cs.utexas.edu

We hypothesize that accounting for the presence of stochasticity in the real world improves sim-to-real transfer of policies learned in simulation. In this work, we analyze the effects of target environment stochasticity in the *sim-to-real* problem. We show several domains where GAT fails to adequately ground the simulator, and we propose a new algorithm, *Stochastic Grounded Action Transformation* (SGAT), that handles this issue gracefully by learning the stochasticity in the environment. We first show on the *Cliff Walking* domain that a policy learned using SGAT achieves better mean returns than GAT on the target environment. We further conduct sim-to-"real" transfer experiments on OpenAI gym MuJoCo environments *InvertedPendulum* and *HalfCheetah*, using a modified version of the environment as a surrogate for the real world.

To test the efficacy of SGAT in handling real world stochasticity, we set up a sim-to-real experiment where a NAO humanoid robot learns to walk on uneven ground. Confirming our hypothesis, we indeed found that policies learned with SGAT, unlike those learned by GAT, learned to cope with environment stochasticity leading to better performance in the real world. Using SGAT, the NAO completed the course 9 out of 10 times (Fig. 1), whereas with GAT, it fell down every time.

II. BACKGROUND

In this section, we review existing literature on state-of-the-art sim-to-real algorithms that are relevant to our approach, and we formalize the problem addressed in this paper.

A. Related Work

In the past, many sim-to-real techniques [4], [7], [8], [9], [10], [11] have achieved policy learning for the real world using a simulator. We can roughly divide these techniques into black-box techniques (which do not modify the simulator internally) and white-box techniques (which do). Another useful distinction is robustness methods vs. grounding methods.

Robustness methods learn a policy on the simulator that is robust to variations in the environment. Dynamics Randomization (DR) is a relatively simple approach that has been shown to be successful at robot control tasks. Peng et al. [7] introduce a white-box DR algorithm that can learn robust RL control policies for robotic manipulation tasks. DR approaches where the effect of actuators (actions) or sensor information (observations) are perturbed with an "envelope" of noise can be considered black-box approaches [8], [9],

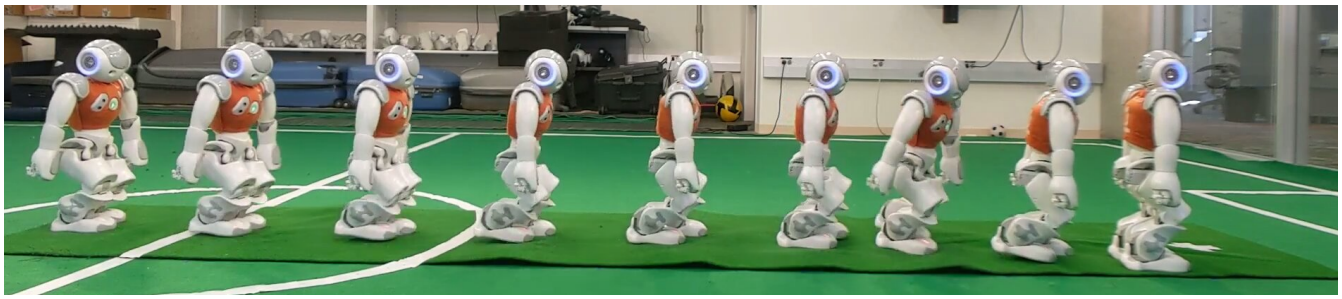


Fig. 1: Experiment setup showing a robot walking on the uneven ground. The NAO *begins walking 40 cms behind the center of the circle and walks 300 cms towards the white penalty cross. This image shows a successful walk executed by the robot at 2 sec intervals, learned using the proposed* SGAT *algorithm.*

[10]. Other DR methods where the internal simulation parameters are perturbed fall under white-box approaches [4], [7], [12], [13]. Robust Adversarial Reinforcement Learning (RARL) [14] randomizes the training environment using an adversary to incorporate robustness into the policy. While DR methods randomly sample simulation parameters before a trajectory is generated, the trajectory is still generated from a deterministic simulator. Our approach models real world stochasticity during individual state transitions.

Orthogonal to robustness methods are approaches that focus on grounding a simulator to behave like a target real world environment. Unlike DR methods that learn a more general robust policy to perform well in any environment, grounding methods learn a policy in a grounded simulator to optimize performance in that specific target environment. Grounded Simulation Learning (GSL) [3] is a learning framework in which data from the real world is used to modify (ground) the simulator. After this grounding step, transitions in the grounded simulator look similar to transitions in the real world environment, hence minimizing the "reality gap". In GSL, all of the learning happens in the simulator; the robot is used only to evaluate policies and to collect transition data [3], [6]. Farchy et al. [3] demonstrate its success at transferring a walk policy from a simulated NAO in the SimSpark simulator to a real SoftBank NAO and achieved 26.7% faster walk than baseline methods.

Based on the GSL framework, the algorithm Grounded Action Transformation (GAT) [6] was introduced recently by Hanna and Stone. Like GSL, GAT grounds the simulator with data from both the real world and the simulator. Policy learning happens only in the grounded simulator; interaction with the real world is required only to evaluate policies and to collect data for grounding the simulator. GAT is a black-box approach to sim-to-real. GAT was experimentally shown to be effective at learning a fast walk policy by grounding a high fidelity simulator, which when deployed in the real world achieved the fastest known walk on the NAO robot: over 43% faster than the state-of-the-art handcoded gait upon which it was based.

B. Preliminaries

We consider the real world (real) and simulation (sim) domains to be two different Markov Decision Processes (MDPs) [15], $\mathcal{M}_{real}$ and $\mathcal{M}_{sim}$ respectively. An MDP comprises a set of states, $\mathcal{S}$, a set of actions, $\mathcal{A}$, the transition dynamics associated with those actions, $\mathcal{T}$, and a reward function, R. At each time step, t, an agent observes the current state, $s_t \in \mathcal{S}$, and chooses an action, $a_t \in \mathcal{A}$, sampled from its policy, $a_t \sim \pi(\cdot|s_t)$. For a given state and action, there is a distribution of possible next states, from which the environment samples a state—$s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t)$. The reward, in this work, is a function of the action and the next state, $r_{t+1} = R(a_t, s_{t+1})$.

In our formulation of the sim-to-real problem, only the transition dynamics $\mathcal{T}_{sim}$ and $\mathcal{T}_{real}$ differ between the environments. Furthermore, we only consider deterministic simulators in this work, but the real environment may have stochastic transitions. The reinforcement learning (RL) objective is to find a policy that maximizes the expected sum of rewards on the real domain, $\mathbb{E}[\sum_{t=0}^{T} R(a_t, s_{t+1})]$.

C. Grounded Action Transformation (GAT)

GAT follows the GSL framework where one alternates between a *grounding step* and a *policy improvement step*. During the grounding step, the policy remains unchanged and is deployed on both sim and real environments to collect state transition data. The GAT algorithm trains two neural networks—a deterministic forward model $\hat{s}_{t+1} = f_{real}(s_t, a_t)$ and an inverse dynamics model $\hat{a}_t = f_{sim}^{-1}(s_t, s_{t+1})$ of the real and sim environments respectively, using supervised learning. The learned forward and inverse models are composed to form the action transformer function $g(s_t, a_t) = f_{sim}^{-1}(s_t, f_{real}(s_t, a_t))$ that grounds the simulator. The inputs to the action transformer are the current state, s_t, and the action chosen by the agent, $a_t \sim \pi(\cdot|s_t)$. The output is a transformed action, $\hat{a}_t$, that when taken in the simulator,

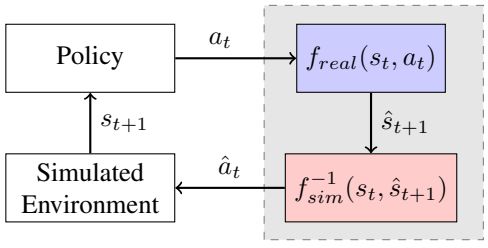
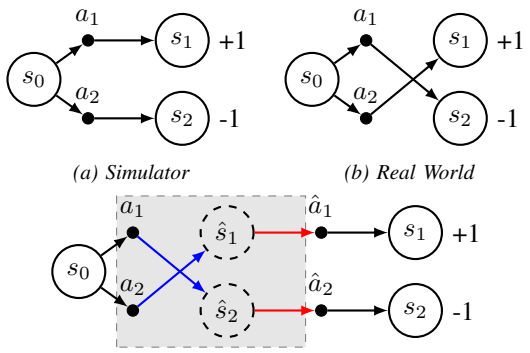


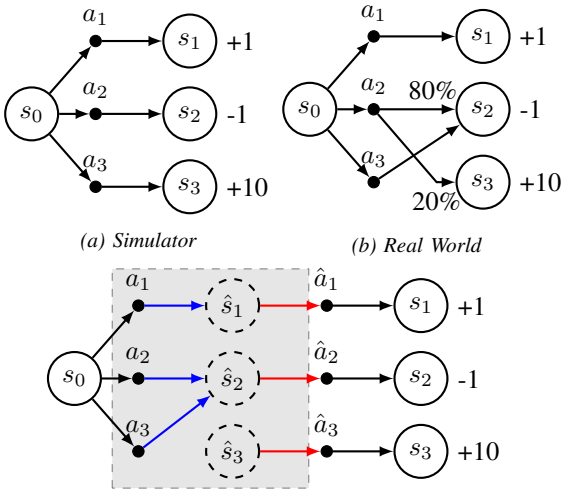
Fig. 2: GAT *Diagram[6]*



(a) Simulator *(b) Real World*

(c) GAT *Grounded Simulator*

Fig. 3: For a deterministic domain, the forward model (blue) and the inverse model (red) can make the grounded simulator behave like the real world. GAT *works well here.*



(a) Simulator *(b) Real World*

(c) GAT *Grounded Simulator*

Fig. 4: When the real world has stochastic transitions, the GAT *forward model (blue) only captures the most likely next state.* GAT *may fail here, since the grounded simulator behaves very differently from the real environment.*

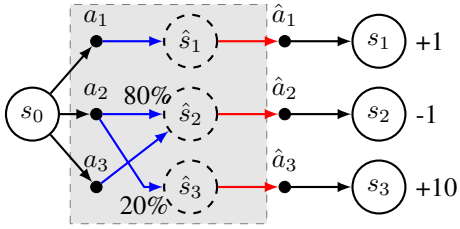


Fig. 5: In the SGAT *Grounded Simulator, the transitions match the real environment (Fig. 4b).*

produces a transition similar to the expected transition when executing a_t in the real world. The combination of the action transformer and the original simulator is known as the grounded simulator. The grounding step is followed by a policy improvement step, in which the parameters of the action transformer are fixed. Thus, the policy improvement step is a standard reinforcement learning problem in the grounded simulator. These two steps are repeated until the optimized policy performs well on the real environment. The block diagram depicting the GAT framework is shown in Fig. 2. While GAT works well on fairly deterministic environments, as was shown by Hanna and Stone [6], in our experimentation, we find that policies learned using GAT perform poorly when transferring to highly stochastic environments.

D. Transferring to a stochastic real world

When the real world domain is deterministic, learning a deterministic forward model, as GAT does, works well. We use a toy example to demonstrate how it works. Consider the environments shown in Figs. 3a and 3b. The agent starts in the initial state, s_0, and chooses between a_1 and a_2. In the simulator, a_1 leads to the higher reward of +1, but the transitions are flipped in reality. Hence, the action a_2 is the optimal action. The GAT action transformer learns to transform a_1 into a_2 and a_2 into a_1. Thus from the agent's perspective the grounded simulator behaves like the real world.

However, when we add stochastic transitions, the two diagrams do not match. In Fig. 4, the optimal action in the simulator is a_3 and in the real world it is a_2; however, in the grounded simulator, it is a_1. Since GAT's forward model is deterministic, it predicts only the most likely next state, whereas other less likely transitions are also important when computing an action's value.

III. STOCHASTIC GROUNDED ACTION TRANSFORMATION (SGAT)

To address real world stochasticity, we introduce Stochastic Grounded Action Transformation (SGAT), which learns a stochastic model of the forward dynamics. In other words, the learned forward model, f_{real}, predicts a distribution over next states, from which we sample, rather than the most likely next state. The sampling operation within the action transformer makes the overall process stochastic. Even though the original simulator is deterministic, the action passed to the action transformation function will not necessarily produce a deterministic effect. Fig. 5 illustrates the simulator from Fig. 4 grounded using SGAT. Since the forward model accounts for stochasticity in the real world, the actions on the grounded simulator have the same effect as the real world.

In continuous state and action domains, we model the next state as a multivariate Gaussian distribution and train the forward model using negative log likelihood (NLL) loss $\mathcal{L} = -\log p(s_{t+1}|s_t, a_t)$. Similar to GAT, we use a neural network function approximator with 2 fully connected hidden layers of 64 neurons to represent the forward and inverse models, but unlike GAT, the forward model in SGAT outputs the parameters of a Gaussian distribution from which we sample the predicted next state.[1] In our implementation, the final dense layer outputs the mean, μ, and the log standard

[1] A Mixture Density Network might be more suitable when the environment's transition dynamics exhibit multimodal behavior.

Algorithm 1 Stochastic Grounded Action Transformation

Input: initial parameters θ_0, ϕ, and ψ for the policy π_θ, forward dynamics model $f_{\phi_{real}}$ and inverse dynamics model $f^{-1}_{\psi_{sim}}$; policy improvement method, `optimize`

1: **while** policy π_θ improves on real **do**
2: Collect real world trajectories $\tau_{real} \leftarrow \{((s_0, a_0), s_1), ((s_1, a_1), s_2), ...\}_{real}$
3: Train forward dynamics function f_ϕ with τ_{real}, using the NLL loss $\mathcal{L} = -log\ p(s_{t+1}|s_t, a_t)$.
4: Collect simulated trajectories $\tau_{sim} \leftarrow \{((s_0, a_0), s_1), ((s_1, a_1), s_2), ...\}_{sim}$
5: Train inverse dynamics function f^{-1}_ϕ with τ_{sim}, using mean squared error loss.
6: Update π_θ on the grounded simulator using `optimize` and the reward from the grounded simulator
7: **end while**

deviation, $log(\sigma)$, for each element of the state vector. The complete algorithm of SGAT is shown in Algorithm 1.

IV. EXPERIMENTS

This section reports on an empirical study of transfer from simulation with SGAT compared to GAT. We begin with a toy reinforcement learning domain and progress to sim-to-real transfer of a bipedal walking controller for a NAO robot. Our empirical results show the benefit of modelling stochasticity when grounding a simulator for transfer to a stochastic real world environment.

A. Cliff Walking

We verify the benefit of SGAT using a classical reinforcement learning domain, the Cliff Walking grid world shown in Fig. 6. In this domain, an agent must navigate around a cliff to reach a goal. The episode terminates when it either reaches the goal (reward of $+100$) or falls into the cliff (reward of -10). There is also a small time penalty (-0.1 per time step), so the agent is incentivized to find the shortest path. There is no discounting, so the agent's objective is to maximize the sum of rewards over an episode. The agent can move up, down, left, or right. If it tries to move into a wall, the action has no effect. In our version of the problem, we assume we have a deterministic simulator, but in the "real" environment, there is a small chance at every time step that the agent moves in a random direction instead of the direction it chose.

Fig. 7 shows GAT and SGAT evaluated for different values of the environment noise parameter. Both the grounding steps and policy improvement steps (using policy iteration [15]) are repeated until convergence for both algorithms. To evaluate the resulting policy, we estimate the expected return from averaging 10,000 episodes. At a value of zero, the "real" environment is completely deterministic. At a value of one, every transition is random. Thus, at both of these endpoints, there is no distinction between the expected return gained by the two algorithms.

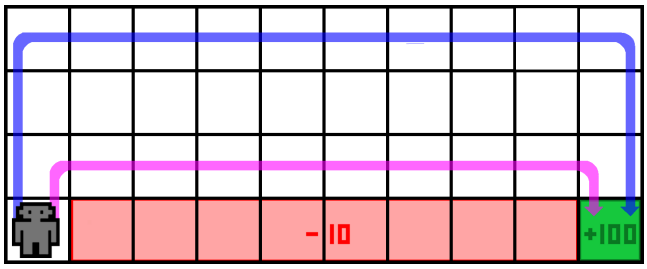


Fig. 6: The agent starts in the bottom left and must reach the goal in the bottom right. Stepping into the red region penalizes the robot and ends the episode. The purple path is the most direct, but the blue path is safer when the transitions are stochastic.

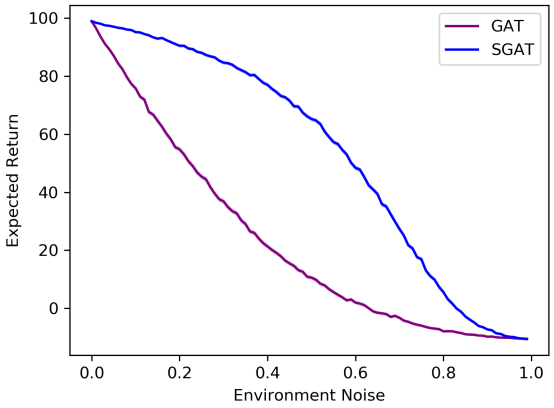


Fig. 7: The y-axis is the average performance of a policy evaluated on the "real" domain. The x-axis is the chance at each time step for the transition to be random. SGAT *outperforms* GAT *for any noise value. Error bars not shown since standard error is smaller than 1 pixel.*

For every intermediate value, SGAT outperforms GAT. The policy trained using GAT is unaware of the stochastic transitions, so it always takes the shortest and most dangerous path. Meanwhile the SGAT agent models the stochasticity, so it learns as if it were training directly on the real environment in the presence of stochasticity.

B. MuJoCo domains

Having shown the efficacy of SGAT in a tabular domain, we now evaluate its performance in continuous control domains that are closer to real world robotics settings. We perform experiments on the OpenAI gym MuJoCo environments to compare the effectiveness of SGAT and GAT when there is added noise in the target domain. We consider the case with just added noise and the case with both noise and domain mismatch between the source and target environments. We call the former Sim-to-NoisySim and the latter Sim-to-NoisyReal. We chose the *InvertedPendulum* and *HalfCheetah* domains to test SGAT in environments with both low and high dimensional state and action spaces. For policy improvement, we use an implementation of Trust Region Policy Optimization (TRPO) [16], from the stable-baselines repository [17].

We simulate stochasticity in the target domains by adding

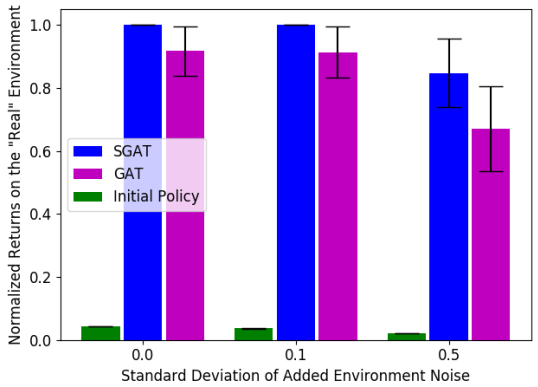


Fig. 8: Performance of best policies learned using GAT *and* SGAT *in a Sim-to-NoisyReal experiment on* ***InvertedPendulum****, averaged over 10 trials. The "real" pendulum is 10 times heavier than the sim pendulum.* SGAT *performs better than* GAT *as the stochasticity in the target domain increases.*

Gaussian noise with different standard deviation values to the actions input into the environment. We omit the results of Sim-to-NoisySim experiments for *InvertedPendulum* because both algorithms performed well on the transfer task. Fig. 8 shows the performance of three different policies on the "real" environment—the initial policy trained in the ungrounded simulator, a policy trained in the grounded simulator using GAT, and a policy trained in the grounded simulator using SGAT. In this Sim-to-NoisyReal experiment, SGAT performs much better than GAT when the stochasticity in the target domain increases. Fig. 9 shows the same experiment on *HalfCheetah*, both with and without domain mismatch. In both Figs. 8 and 9, the returns are normalized with respect to 1.0 being the best return a policy trained in the "real" environment could achieve, and the error bars show the standard error across 10 trials. While policies learned using both algorithms are similar when the stochasticity is minimal in the target domain, as the stochasticity increases, SGAT policies perform better than those learned using GAT.

C. NAO *Robot Experiments*

Until this point in our analysis, we have used a modified version of the simulator in place of the "real" world so as to isolate the effect of stochasticity (as opposed to domain mismatch). However, the true objective of this research is to enable transfer to real robots, which may exhibit very different noise profiles than the simulated environments. Thus, in this section, we validate SGAT on a real humanoid robot learning to walk on uneven terrain.

Our robot experiments were conducted on the SoftBank NAO bipedal robot, used in the RoboCup robot soccer competitions. For learning in simulation, we use the SimSpark physics simulator, used in the RoboCup 3D Simulation league. We compared GAT and SGAT by independently learning control policies using these algorithms to walk on uneven terrain, as shown in Fig. 1. To create uneven ground in the lab, we placed foam packing material under the turf of the robot soccer field. On this uneven ground, the walking

TABLE I: Speed and stability of NAO *robot walking on uneven ground. The initial policy θ_0 walks at 14.66 ± 1.65 cm/s and always falls down. Both* SGAT *and* GAT *find policies that are faster, but* SGAT *policies are more stable than policies learned using* GAT.

	Grounding Step 1		Grounding Step 2	
	Speed (cm/s)	Falls	Speed (cm/s)	Falls
GAT	15.7 ± 2.98	6/10	18.5 ± 3.63	10/10
SGAT	16.9 ± 0.678	**0/10**	18.0 ± 2.15	**1/10**

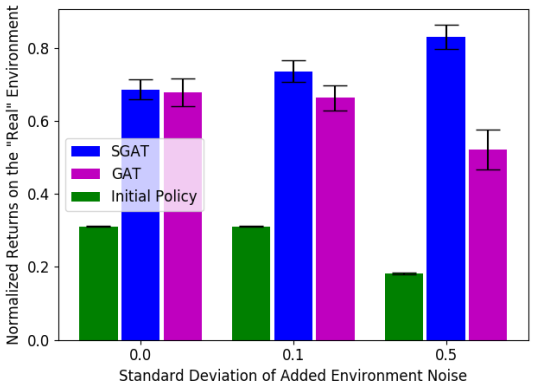
dynamics become more random, since the forces acting on the foot are slightly different every time the robot takes a step. For the walk policy, we use the rUNSWift walk engine and the initial published parameter set θ_0, developed by Ashar et al. [18]. This initial unoptimized policy achieves a speed of 14.66 ± 1.65 cm/s on the uneven terrain. The walk engine contains 16 parameters that we optimize on the grounded simulator using Covariance Matrix Adaptation - Evolutionary Strategy (CMA-ES), with a population size of 150. Each trajectory lasts for 7.5 seconds on the simulator or terminates when the robot falls down. The reward function is a sum of the forward velocity of the robot (in cm/s) and an early termination penalty of -15 if the robot falls down. Thus, CMA-ES optimizes the policy for walks that are faster and more stable in the grounded simulator.

On flat ground, both methods produced very similar policies, but on the uneven ground, the policy learned using SGAT was more successful than a policy learned using GAT. We performed ten trial runs of the best policy learned using SGAT and GAT after each grounding step, and the average speed of the robot on the uneven terrain is shown in Table I. The policy learned using SGAT takes shorter steps and stays upright, thereby maintaining its balance on the uneven terrain, whereas the policy produced using GAT learned to lean forward and walk faster, but fell down more often due to the uneven terrain. This result is best visualized in the supplementary video attached with this paper. Both algorithms produce policies that improve the walking speed across grounding steps. The GAT policy after the second grounding step always falls over, whereas the SGAT policy was more stable and finished the course 9 out of 10 times.

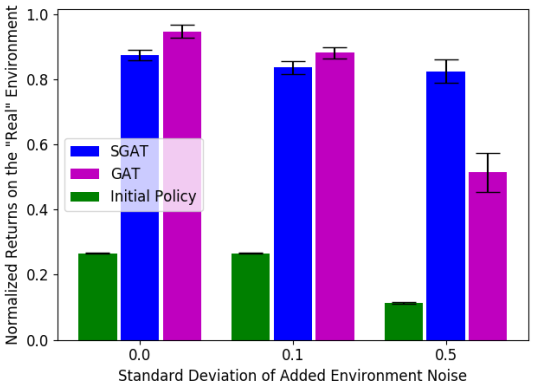
V. DISCUSSION AND LIMITATIONS

In our experiments, we observe that stochastic transitions can have a detrimental effect on how GAT grounds the simulator; however both algorithms perform similarly on deterministic environments. In real world scenarios, we cannot know how stochastic an environment is before testing. This fact suggests that we should default to SGAT.

In this work, we have only considered deterministic simulators, but simulators may have stochastic transitions as well, especially if the simulator was designed to anticipate process noise. However, when using an action transformer grounding approach, stochastic simulators make the learning problem more difficult. We can no longer sample from the distribution provided by the forward model. Instead, the inverse model must take in a distribution over states and output a distribution over actions.



(a) Sim-to-NoisySim



(b) Sim-to-NoisyReal

Fig. 9: Performance of best policies learned using GAT *and* SGAT *in the* ***HalfCheetah*** *domain, analyzed in Sim-to-NoisySim and Sim-to-NoisyReal experiments. In the NoisyReal environment, the "real" HalfCheetah's torso is 15% heavier than the sim HalfCheetah. When the "real" environment is highly stochastic,* SGAT *performs better than* GAT.

VI. CONCLUSION AND FUTURE WORK

In this work, we introduced *Stochastic Grounded Action Transformation* (SGAT), a sim-to-real algorithm, which gracefully handles policy learning in simulation when the target domain is stochastic.

First, in *Cliff Walking*, we empirically showed that by accounting for target domain stochasticity, SGAT policies can produce much better average returns on the real environment than GAT policies. Then, we compared the two algorithms on two MuJoCo environments in Sim-to-NoisySim and Sim-to-NoisyReal scenarios and showed that SGAT policies perform better when the target environment is highly stochastic. To verify the algorithm in a real world transfer setting, we conducted an experiment in which a NAO humanoid robot learns to walk over uneven terrain. The empirical results indicate that SGAT policies are more robust to environmental stochasticity and walk more robustly over the uneven terrain compared to hand-coded polices and GAT policies.

Though we have focused on black-box techniques, these ideas may apply to some white-box techniques as well. Many sim-to-real algorithms are designed to find correct environment parameters, but the most accurate model might be one in which these parameters are sampled from a distribution. Investigating this scenario is a promising direction for future work.

VII. ACKNOWLEDGMENTS

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CPS-1739964, IIS-1724157, NRI-1925082), ONR (N00014-18-2243), FLI (RFP2-000), ARL, DARPA, Lockheed Martin, GM, and Bosch. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

REFERENCES

[1] M. Cutler and J. P. How, "Efficient reinforcement learning for robots using informative simulated priors," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2605–2612, 2015.

[2] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, pp. 1238 – 1274, 2012.

[3] A. Farchy, S. Barrett, P. MacAlpine, and P. Stone, "Humanoid robots learning to walk faster: From the real world to simulation and back," in *Proc. of 12th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2013.

[4] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *Robotics: Science and Systems XIV*, Jun 2018. [Online]. Available: http://dx.doi.org/10.15607/RSS.2018.XIV.010

[5] F. Golemo, A. A. Taiga, A. Courville, and P.-Y. Oudeyer, "Sim-to-real transfer with neural-augmented robot simulation," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 817–828. [Online]. Available: http://proceedings.mlr.press/v87/golemo18a.html

[6] J. Hanna and P. Stone, "Grounded action transformation for robot learning in simulation," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, February 2017.

[7] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *CoRR*, vol. abs/1710.06537, 2017. [Online]. Available: http://arxiv.org/abs/1710.06537

[8] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *Advances in Artificial Life*, F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 704–720.

[9] O. Miglino, H. H. Lund, and S. Nolfi, "Evolving mobile robots in simulated and real environments," *Artificial Life*, vol. 2, no. 4, pp. 417–434, 1995. [Online]. Available: https://doi.org/10.1162/artl.1995.2.4.417

[10] ——, "Evolving mobile robots in simulated and real environments," *ARTIFICIAL LIFE*, vol. 2, pp. 417–434, 1996.

[11] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "Epopt: Learning robust neural network policies using model ensembles," 2016.

[12] A. Molchanov, T. Chen, W. Honig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019. [Online]. Available: http://dx.doi.org/10.1109/IROS40897.2019.8967695

[13] W. Yu, J. Tan, C. Karen Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system

identification," *Robotics: Science and Systems XIII*, Jul 2017. [Online]. Available: http://dx.doi.org/10.15607/RSS.2017.XIII.048

[14] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," 2017.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[16] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," 2015.

[17] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[18] J. Ashar, J. Ashmore, B. Hall, S. Harris, B. Hengst, R. Liu, Z. Mei, M. Pagnucco, R. Roy, C. Sammut, O. Sushkov, B. Teh, and L. Tsekouras, "Robocup spl 2014 champion team paper," vol. 8992, pp. 70–81, 01 2015.