Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

# Grounded Action Transformation for Robot Learning in Simulation

**Josiah P. Hanna** and **Peter Stone**
{jphanna,pstone}@cs.utexas.edu
The University of Texas at Austin
2317 Speedway, Austin, TX 78712 USA

## Abstract

Robot learning in simulation is a promising alternative to the prohibitive sample cost of learning in the physical world. Unfortunately, policies learned in simulation often perform worse than hand-coded policies when applied on the physical robot. This paper proposes a new algorithm for learning in simulation – Grounded Action Transformation – and applies it to learning of humanoid bipedal locomotion. Our approach results in a 43.27% improvement in forward walk velocity compared to a state-of-the art hand-coded walk.[1]

## Introduction

An alternative to learning robotic skills directly on the physical robot is learning in simulation and transferring the learned skills. However, the value of simulation learning is limited by the inherent inaccuracy of simulators in modeling the dynamics of the physical world. As a result, learning that takes place in a simulator is unlikely to increase real world performance.

Grounded Simulation Learning (GSL) is a framework for modifying a simulator such that learning transfers to the physical robot (Farchy et al. 2013). We introduce a new GSL algorithm – **G**rounded **A**ction **T**ransformation (GAT) for simulation-transfer. The algorithm is evaluated on the task of bipedal robotic walking. Our evaluation of GAT starts from a state-of-the-art walk engine as the base policy and improves the walk velocity by over 43%, leading to what may be the fastest known walk on the SoftBank NAO robot.

## Problem Statement

At discrete time-step $t$ the robot takes action $A_t \sim \pi(\cdot|S_t)$ according to $\pi$ which is a distribution over actions, $A_t \in \mathcal{A}$, conditioned on the current state, $S_t \in \mathcal{S}$. The environment, $E$, responds with $S_{t+1} \sim P(\cdot|S_t, A_t)$ according to the dynamics, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ which is a probability density function over next states conditioned on the current state and action. A trajectory of length $L$ is a sequence of states and actions, $\tau := S_0, A_0, ..., S_L, A_L$. The cost function, $c$, assigns a scalar cost to each $(s, a)$ pair. The value of policy, $\pi$, is defined as $J(\pi) := \mathbb{E}_{\tau \sim Pr(\cdot|\pi)} \left[ \sum_{t=0}^{L} c(S_t, A_t) \right]$ where $Pr(\tau|\pi)$ is the probability of $\tau$ when selecting actions according to $\pi$. The policy $\pi$ is parameterized by a vector $\boldsymbol{\theta}$. Since $\boldsymbol{\theta}$ determines the policy distribution we overload notation and refer to $\pi_{\boldsymbol{\theta}}$ as $\boldsymbol{\theta}$. Given initial parameters $\boldsymbol{\theta}_0$, the goal of policy improvement is to find $\boldsymbol{\theta}'$ such that $J(\boldsymbol{\theta}') < J(\boldsymbol{\theta}_0)$.

In this paper, learning takes place in a simulator, $E_{\texttt{sim}}$, that models $E$. Specifically $E_{\texttt{sim}}$ has the same state-action space as $E$ but inevitably a different dynamics distribution, $P_{\texttt{sim}}$. The different dynamics distribution mean that for any policy, $\boldsymbol{\theta}$, $J(\boldsymbol{\theta}) \neq J_{\texttt{sim}}(\boldsymbol{\theta})$ since $\boldsymbol{\theta}$ induces a different trajectory distribution in $E$ than in $E_{\texttt{sim}}$. Thus $\boldsymbol{\theta}'$ with $J_{\texttt{sim}}(\boldsymbol{\theta}') < J_{\texttt{sim}}(\boldsymbol{\theta}_0)$ does not imply $J(\boldsymbol{\theta}') < J(\boldsymbol{\theta}_0)$ – in fact $J(\boldsymbol{\theta}')$ could even be worse than $J(\boldsymbol{\theta}_0)$. This paper explores methods for learning in $E_{\texttt{sim}}$ that lower expected cost.

## Grounded Action Transformation

The Grounded Simulation Learning (GSL) framework (Farchy et al. 2013) improves robot learning in simulation by using state transition data from the physical system to ground $E_{\texttt{sim}}$ such that it is a better model of $E$. The GSL framework assumes there is an imperfect simulator, $E_{\texttt{sim}} = \langle \mathcal{S}, \mathcal{A}, P_{\texttt{sim}}, c \rangle$, that models $E$. Furthermore, $E_{\texttt{sim}}$ must be *modifiable*. A modifiable simulator has parameterized transition probabilities $P_{\boldsymbol{\phi}}(\cdot|s, a) := P_{\texttt{sim}}(\cdot|s, a; \boldsymbol{\phi})$ where the vector $\boldsymbol{\phi}$ can be changed to produce, in effect, a different simulator.

GSL grounds $E_{\texttt{sim}}$ by finding $\boldsymbol{\phi}^\star$ such that:

$$\boldsymbol{\phi}^\star = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \sum_{\tau \in \mathcal{D}} d\left(Pr(\tau|\boldsymbol{\theta}), Pr_{\texttt{sim}}(\tau|\boldsymbol{\theta}, \boldsymbol{\phi})\right) \quad (1)$$

where $Pr(\tau|\boldsymbol{\theta})$ is the probability of observing trajectory $\tau$ in $E$, $Pr_{\texttt{sim}}(\tau|\boldsymbol{\theta}, \boldsymbol{\phi})$ is the probability of $\tau$ in $E_{\texttt{sim}}$ with dynamics parameterized by $\boldsymbol{\phi}$, $d$ is a measure of similarity between these probabilities, and $\mathcal{D}$ is a data set of trajectories recorded on the physical robot. When a policy improvement method is used to optimize $\boldsymbol{\theta}$ in $E_{\texttt{sim}}$ with $\boldsymbol{\phi}^\star$ the improved policy, $\boldsymbol{\theta}'$ is expected to perform better on the physical robot.

Our principle algorithmic contribution, GSL with *Grounded Action Transformation* (GAT) improves grounding by correcting differences in the effects of actions between



[1] An extended version of this work is published in AAAI 2017. Videos of learned walk policies can be found at https://www.cs.utexas.edu/users/AustinVilla/?p=research/real_and_sim_walk_learning.

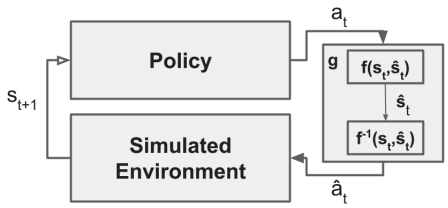


Figure 1: GAT induces a modifiable simulator.

$E$ and $E_{\texttt{sim}}$. Since it is often easier to minimize error in the one step dynamics distribution, GAT uses:

$$\boldsymbol{\phi}^{\star} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \sum_{\tau_i \in \mathcal{D}} \sum_{t=0}^{L} d(P(\mathbf{s}^i_{t+1}|\mathbf{s}^i_t, \mathbf{a}^i_t), P_{\boldsymbol{\phi}}(\mathbf{s}^i_{t+1}|\mathbf{s}^i_t, \mathbf{a}^i_t)) \tag{2}$$

as a surrogate loss function for (1) which can be minimized with transitions observed in $\mathcal{D}$.

To find $\boldsymbol{\phi}^{\star}$ efficiently, GAT uses a parameterized *action transformation* function which takes the agent's state and action as input and outputs a new action which – when taken in simulation – will result in the robot transitioning to the same next state it would have in $E$. We denote this function, $g_{\boldsymbol{\phi}} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{A}$; the parameters of $g$ serve as $\boldsymbol{\phi}$ under the GSL framework. GAT constructs $g$ with parameterized models of the robot's dynamics and the simulators *inverse* dynamics. Assuming a dataset $\mathcal{D}_{\texttt{sim}}$ of simulated experience, GAT reduces (2) to a supervised learning problem.

GAT defines $g \coloneqq g_{\boldsymbol{\phi}}$ by a deterministic forward model of the robot's dynamics, $f$ and a deterministic model of the simulator's inverse dynamics, $f^{-1}_{\texttt{sim}}$. Let $\mathbf{x}_t$ be the robot's joint configuration at time $t$. The function, $f$ maps $(\mathbf{s}_t, \mathbf{a}_t)$ to the maximum likelihood estimate of $\mathbf{x}_{t+1}$ under $P$. The function $f^{-1}_{\texttt{sim}}$ maps $(\mathbf{s}_t, \mathbf{s}_{t+1})$ to the action that is most likely to produce this transition in simulation. When executing $\boldsymbol{\theta}$ in simulation, the robot selects $\mathbf{a}_t \sim \pi_{\boldsymbol{\theta}}(\cdot|\mathbf{s}_t)$ and then uses $f$ to predict what the resulting configuration, $\mathbf{x}_{t+1}$, would be in $E$. Then $\mathbf{a}_t$ is replaced with $\hat{\mathbf{a}}_t \coloneqq f^{-1}_{\texttt{sim}}(\mathbf{s}_t, f(\mathbf{s}_t, \mathbf{a}_t))$. The result is the robot achieves the exact $\mathbf{x}_{t+1}$ it would have on the physical robot. In practice $f$ and $f^{-1}_{\texttt{sim}}$ are represented with supervised regression models (neural networks in this work) and learned from $\mathcal{D}$ and $\mathcal{D}_{\texttt{sim}}$ respectively. Figure 1 illustrates the GAT-modified $E_{\texttt{sim}}$. GAT then proceeds to improve $\boldsymbol{\theta}$ with `optimize` within the grounded simulator.

## Empirical Results

We evaluate GAT on the task of bipedal robot walking using the SoftBank NAO. For walking, our NAO uses an open source walk engine developed at the University of New South Wales (UNSW) (Ashar et al. 2015). This walk engine has been used by at least one team in each of the past three RoboCup Standard Platform League (SPL) championship games in which teams of five NAOs compete in soccer matches. On the physical robot a trajectory terminates once the robot has walked four meters or falls. A trajectory generated with $\boldsymbol{\theta}_0$ lasts $\approx 20.5$ seconds on the robot. We use two different simulators in this work: the open source SimSpark and Gazebo simulators. In simulation a trajectory terminates after a fixed time. We use the Covariance Matrix Adaption-Evolutionary Strategy (CMA-ES) algorithm (Hansen 2006) for policy improvement within simulation. To clarify terminology, a generation refers to a single update of CMA-ES; an iteration refers to a complete cycle of GSL.

| $\boldsymbol{\theta}_0$ | GAT with SimSpark | GAT with Gazebo |
|---|---|---|
| 19.52 (cm/s) | 26.27 | 26.89 |
| 0.0 (%) | 34.58 | 37.76 |

Table 1: This table gives the maximum learned velocity and percent improvement over $\boldsymbol{\theta}_0$ (left column).

We evaluate GAT for transferring simulation learning to the physical NAO using both SimSpark and Gazebo as $E_{\texttt{sim}}$. The data set $\mathcal{D}$ consists of 15 trajectories collected with $\boldsymbol{\theta}_0$ on the physical NAO. For each iteration, we optimize $\boldsymbol{\theta}$ for 10 generations of the CMA-ES algorithm. We evaluate the best policy from each generation with 5 trajectories on the physical robot. If the robot falls in any of the 5 trajectories the policy is considered unstable.

### Experimental Results

Table 1 gives the walk velocity of the best $\boldsymbol{\theta}'$ returned by GAT in each simulator. GAT with SimSpark and GAT with Gazebo both improved walk velocity by over 30%. This result demonstrates generality of GAT across different simulators. Two iterations of GAT with SimSpark (not shown) increased the walk velocity of the NAO by 43.27% compared to $\boldsymbol{\theta}_0$.

Policy improvement with CMA-ES required 30,000 trajectories per iteration to find the 10 policies that were evaluated on the robot. In contrast the total number of trajectories executed on the physical robot is 65 (15 trajectories in $\mathcal{D}$ and 5 evaluations per $\boldsymbol{\theta}'$). This result demonstrates GAT can use sample-intensive simulation learning to optimize real world skills with a low number of trajectories on the physical robot.

## Conclusion

We proposed and evaluated the Grounded Action Transformation (GAT) algorithm for grounded simulation learning. Our method led to a 43.27 % improvement in the walk velocity of a state-of-the-art bipedal robot. Furthermore, this experiment demonstrated the benefits of GAT are independent of the choice of simulator.

## References

Ashar, J.; Ashmore, J.; Hall, B.; and Harris, S. e. a. 2015. Robocup spl 2014 champion team paper. In *RoboCup 2014: Robot World Cup XVIII*, volume 8992 of *Lecture Notes in Computer Science*. Springer International Publishing. 70–81.

Farchy, A.; Barrett, S.; MacAlpine, P.; and Stone, P. 2013. Humanoid robots learning to walk faster: From the real world to simulation and back. In *Twelth International Conference on Autonomous Agents and Multiagent Systems, AAMAS*.

Hansen, N. 2006. The CMA evolution strategy: a comparing review. In Lozano, J.; Larranaga, P.; Inza, I.; and Bengoetxea, E., eds., *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*. Springer. 75–102.