Grounded Action Transformation for Robot Learning in Simulation

Josiah Hanna and Peter Stone
Reinforcement Learning for Physical Robots

Learning on physical robots:

- Not data-efficient.
- Requires supervision.
- Manual resets.
- Robots break.
- Wear and tear make learning non-stationary.

Not an exhaustive list...
Learning in simulation:

- Thousands of trials in parallel.
- No supervision and automatic resets.
- Robots never break or wear out.
Learning in simulation:

- Thousands of trials in parallel.
- No supervision and automatic resets.
- Robots never break or wear out.

Policies learned in simulation often fail in the real world.
Notation

Environment \( E = \langle S, A, c, P \rangle \)
- Robot in state \( s \in S \) chooses action \( a \in A \) according to policy \( \pi \).
  - Parameterized \( \pi_\theta \) denoted \( \theta \)
- Environment, \( E \), responds with a new state \( S_{t+1} \sim P(\cdot|s, a) \).
- Cost function \( c \) defines a scalar cost for each \( (s, a) \).
- Goal is to find \( \theta \) which minimizes:

\[
J(\theta) := \mathbb{E}_{S_1, A_1, \ldots, S_L, A_L} \left[ \sum_{t=1}^{L} c(S_t, A_t) \right]
\]
Learning in Simulation

Simulator $E_{\text{sim}} = \langle S, A, c, P_{\text{sim}} \rangle$.

- Identical to $E$ but different dynamics (transition function).
Simulator $E_{\text{sim}} = \langle S, A, c, P_{\text{sim}} \rangle$.

- Identical to $E$ but different dynamics (transition function).

\[ J_{\text{sim}}(\theta') > J_{\text{sim}}(\theta_0) \not\Rightarrow J(\theta') > J(\theta_0) \]

Goal: Learn $\theta$ in simulation that also works on physical robot.
Grounded Simulation Learning (GSL) is a framework for robot learning in simulation by modifying the simulator with real world data so that policies learned in simulation work in the real world [?].

1. Execute $\theta_0$ on physical robot.
2. Ground simulator so $\theta_0$ produces similar trajectories in simulation.
3. Optimize $J_{\text{sim}}(\theta)$ to find better $\theta'$.
4. Test $\theta'$ on the physical robot.
5. $\theta_0 := \theta'$ and repeat.
Grounded Simulation Learning

1. Real world state-action trajectories
2. Real World Policy Execution
3. Improved Policy
4. Simulate Grounding
5. Grounded Simulator
6. Policy Improvement in Simulation
Grounding the Simulator

Assume $P_{\text{sim}}$ is parameterized by $\phi$.

d: Any measure of similarity between state transition distributions

Robot executes $\theta_0$ and records dataset $\mathcal{D}$ of $(S_t, A_t, S_{t+1})$ transitions.

$$\phi^* = \arg\min_{\phi} \sum_{(S_t, A_t, S_{t+1}) \in \mathcal{D}} d(P(\cdot \mid S_t, A_t), P_\phi(\cdot \mid S_t, A_t))$$
Grounding the Simulator

Assume $P_{\text{sim}}$ is parameterized by $\phi$.

$d$: Any measure of similarity between state transition distributions

Robot executes $\theta_0$ and records dataset $\mathcal{D}$ of $(S_t, A_t, S_{t+1})$ transitions.

$$\phi^* = \arg\min_{\phi} \sum_{(S_t, A_t, S_{t+1}) \in \mathcal{D}} d(P(\cdot | S_t, A_t), P_\phi(\cdot | S_t, A_t))$$

How to define $\phi$?
Advantages of GSL

1. No random-access simulation modification required.
2. Leaves underlying policy optimization unchanged.
3. Efficient simulator modification.
Farchy et al. presented a GSL algorithm and demonstrated a 26.7% improvement in walk speed on a Nao.

Two limitations of existing approach:

1. Modification relied on assumption that desired joint positions achieved instantaneously in simulation.

2. Used expert knowledge to select which components of $\theta$ could be learned.
Grounded Action Transformations

Goal: Eliminate simulator-dependent assumption of earlier work.

\[
\phi^* = \arg\min_{\phi} \sum_{(S_t, A_t, S_{t+1}) \in \mathcal{D}} d(P(\cdot | S_t, A_t), P_{\phi}(\cdot | S_t, A_t))
\]

Replace robot’s action \(a_t\) with an action that produces a more “realistic” transition.
Learn this action as a function \(g_{\phi}(s_t, a_t)\).
Grounded Action Transformation

Figure: Modifiable simulator induced by GAT.
\( \mathcal{X} \): the set of robot joint configurations.

Learn two functions:

- Robot’s dynamics: \( f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{X} \)
- Simulator’s inverse dynamics: \( f_{\text{sim}}^{-1} : \mathcal{S} \times \mathcal{X} \rightarrow \mathcal{A} \).

Replace robot’s action \( a_t \) with \( \hat{a}_t := f_{\text{sim}}^{-1}(s_t, f(s_t, a_t)) \).
Grounded Action Transformations

Figure: Modifiable simulator induced by GAT.
GAT Implementation

\( f \) and \( f^{-1}_{\text{sim}} \) learned with supervised learning.

- Record sequence \( S_t, A_t, \ldots \) on robot and in simulation.
- Supervised learning of \( g \):
  - \( f^{-1}_{\text{sim}} : (S_t, A_t) \rightarrow X_{t+1} \)
  - \( f : (S_t, X_{t+1}) \rightarrow A_t \)

Smooth modified actions:

\[
g(s_t, a_t) := \alpha f^{-1}_{\text{sim}}(s_t, f(s_t, a_t)) + (1 - \alpha)a_t
\]
Supervised Implementation

- Forward model trained with 15 real world trajectories of 2000 time-steps.
- Inverse model trained with 50 simulated trajectories of 1000 time-steps.
Applied GAT to learning fast bipedal walks for the Nao robot.

- Task: Walk forward towards a target.
- $\theta_0$: University of New South Wales Walk Engine.
- Simulator: SimSpark Robocup3D Simulator and OSRF Gazebo Simulator.
- Policy optimization with CMA-ES stochastic search method.
Empirical Results

(a) Softbank Nao  
(b) Gazebo Nao  
(c) SimSpark Nao
Empirical Results
## Empirical Results

### Simulation to Nao:

<table>
<thead>
<tr>
<th>Method</th>
<th>Velocity (cm/s)</th>
<th>% Improve</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial policy</td>
<td>19.52</td>
<td>0.0</td>
</tr>
<tr>
<td>SimSpark, first iteration</td>
<td>26.27</td>
<td>34.58</td>
</tr>
<tr>
<td>SimSpark, second iteration</td>
<td>27.97</td>
<td>43.27</td>
</tr>
<tr>
<td>Gazebo, first iteration</td>
<td>26.89</td>
<td>37.76</td>
</tr>
</tbody>
</table>

### SimSpark to Gazebo:

<table>
<thead>
<tr>
<th>Method</th>
<th>% Improve</th>
<th>Failures</th>
<th>Best Gen.</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Ground</td>
<td>11.094</td>
<td>7</td>
<td>1.33</td>
</tr>
<tr>
<td>Noise-Envelope</td>
<td>18.93</td>
<td>5</td>
<td>6.6</td>
</tr>
<tr>
<td>GAT</td>
<td><strong>22.48</strong></td>
<td><strong>1</strong></td>
<td><strong>2.67</strong></td>
</tr>
</tbody>
</table>
Conclusion

Contributions:

1. Introduced Grounded Action Transformations algorithm for simulation transfer.

2. Improved walk speed of Nao robot by over 40 % compared to state-of-the-art walk engine.

Future Work:

- Extending to other robotics tasks and platforms.
- When does grounding actions work and when does it not?
- Reformulating learning $g$:
  - $f$ and $f_{\text{sim}}^{-1}$ minimize one-step error but we actually care about error over sequences of states and actions.
Thanks for your attention!
Questions?
Alon Farchy, Samuel Barrett, Patrick MacAlpine, and Peter Stone.

Humanoid robots learning to walk faster: From the real world to simulation and back.