Reinforcement Learning Project: Open AI Robotic Hand (Robby)

Patrick Matthäi & Andrea Maldonado

Task

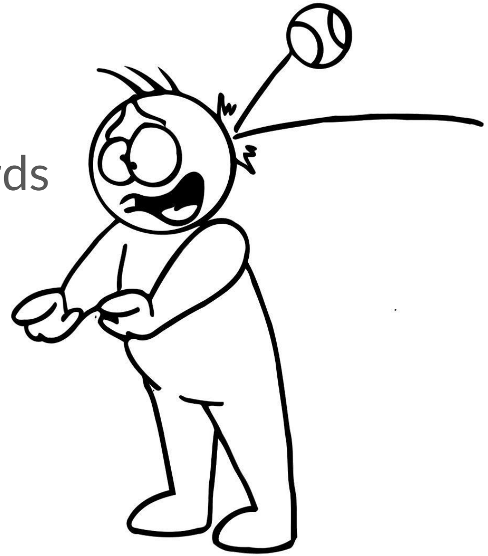
Train a simulated robot hand to throw a ball as high as possible

Challenges:

- Project setups (Licensing)
- Creating a new environment for the task
- Rewards
- Learning an effective throw motion from a vast search space

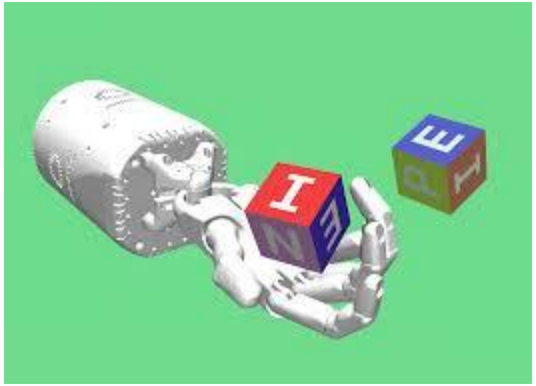
Sparse Reward Problem’s Challenges

- Promoting a throwing motion
  - Losing ball’s touch is losing control
  - Gravity and ball max height as goal:
    height and velocity after turning point
  - Preventing the Cobra Effect when designing Rewards



Setup

- **Simulation/Physic Engine:** MuJoCo
- **Environment**: gym + mujoco-py
- **Python** 3.7
- **Frameworks**: tensorflow, tflearn, matplotlib
- **Dependencies**: pydmps, deep-rl

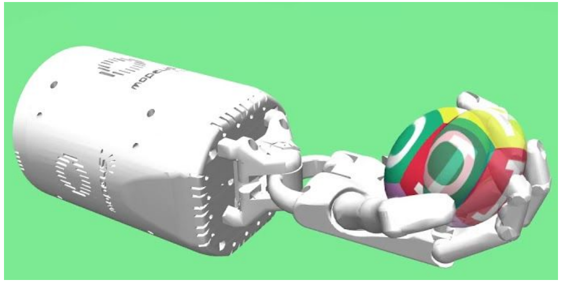


matplotlib

OpenAI

TensorFlow

Environment and agent changes

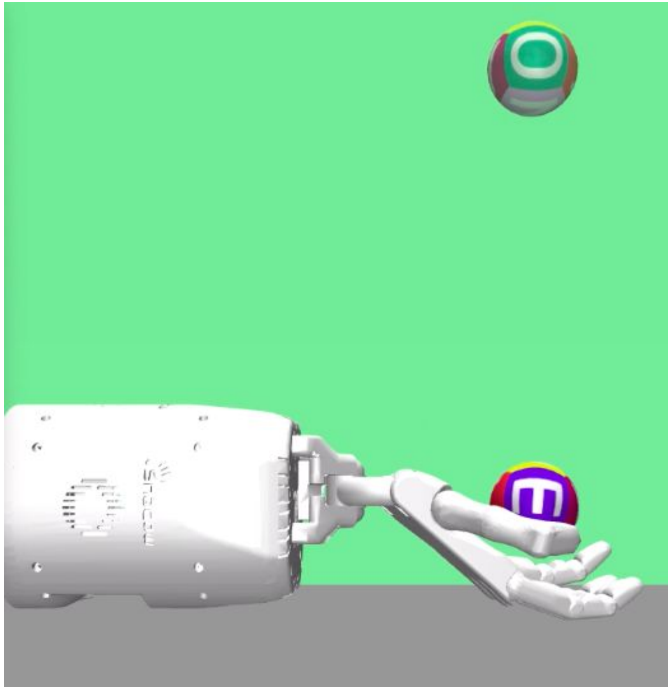


- **Source Environment and agent:**
  Open AI Robotics ⇒ “HandManipulateEgg-v0”

- **Changes:**
  - Collision detection between ball and floor ⇒ Reset environment
  - Mobility of hand joints (actuation range, damping)
  - Lighter ball (reduce mass from 1 to 0.1)
  - New reward function (ball height, ball velocity)
  - Visual changes (moved camera, floor-Mesh, coloring)

Reward function

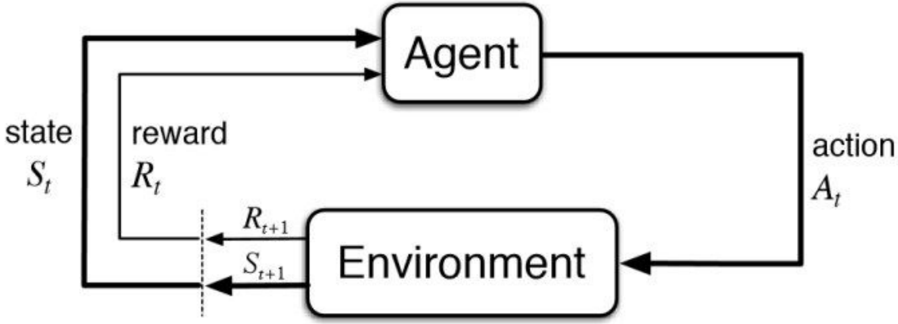
Ball height
Ball velocity on height axis
Ball distance to target
Depenalizing falling



Environment and agent

Action space:

- Rotation of a subset of hand joints
- 20 Dimensions



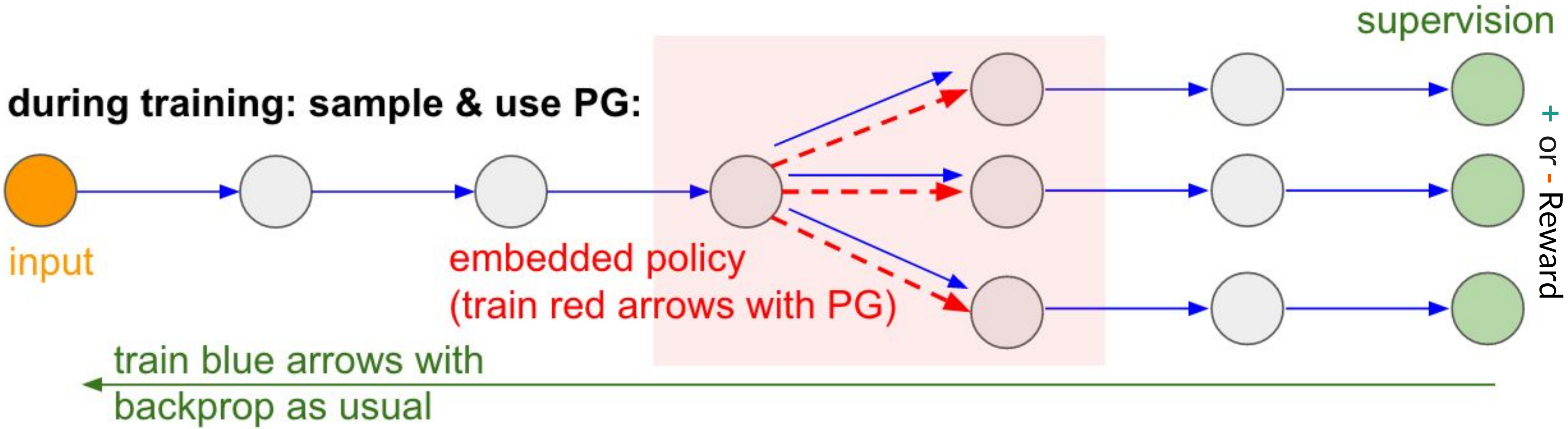
State space:

- Hand: Positions of all hand joints and velocities in all dimensions
- Ball: Positions and positional velocities in all dimensions
- Goal: Positions and positional velocities in all dimensions
- 54 Dimensions

Episode

1. Place ball in palm
2. Start in a given inertial position
3. Move joints by one of our solutions
    a. If ball collides with floor → reset environment and start at 1.

Policy Gradient Methods

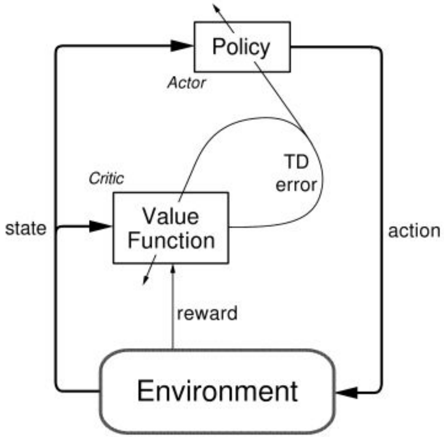


Deep Deterministic Policy Gradient

- Model-free, **off-policy**, actor-critic learning algorithm

  Goal: Represent the policy function independently of the value function.

- **Used utilities:** Q-learning, replay buffer, batch normalization, mini batch learning, actor noise



Deterministic Policy Gradient Theorem

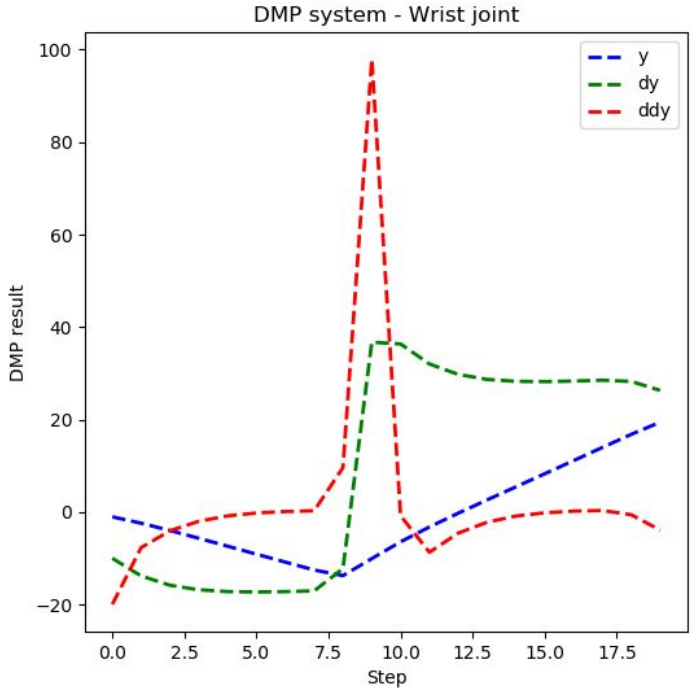
$$\nabla_{\theta^\mu} \mu \approx \mathbb{E}_{\mu'} \left[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_t} \right]$$

Dynamic Movement Primitives

- **Goal**: Reduce the search space

- Reducing freedom
- Dividing complex into simple
- Trajectory to emulate throwing motion for position,velocity and acceleration.

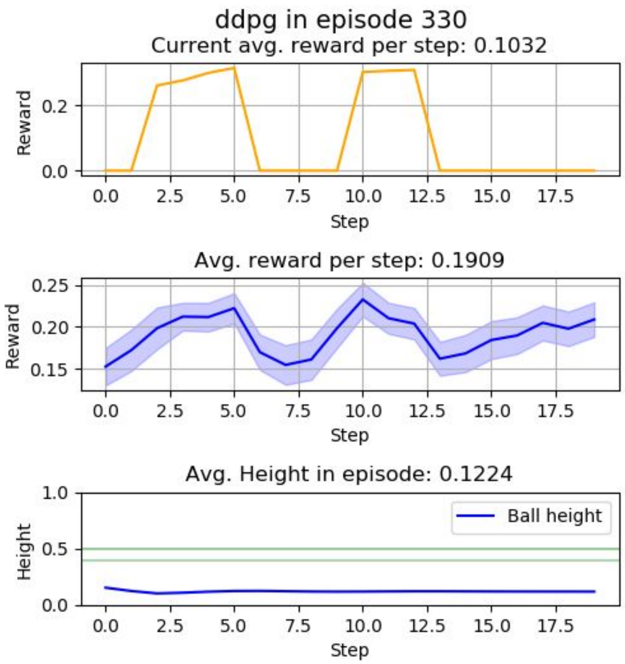
- 200 basis functions and τ = 5
  $\Rightarrow$ 20 steps lasting throwing motion

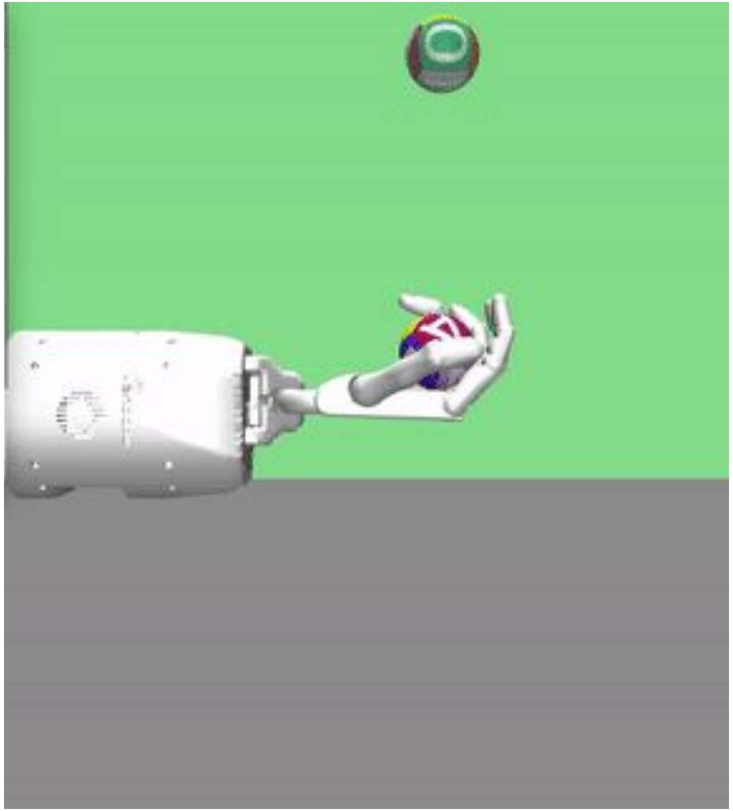


Combining DDPG with DMPs

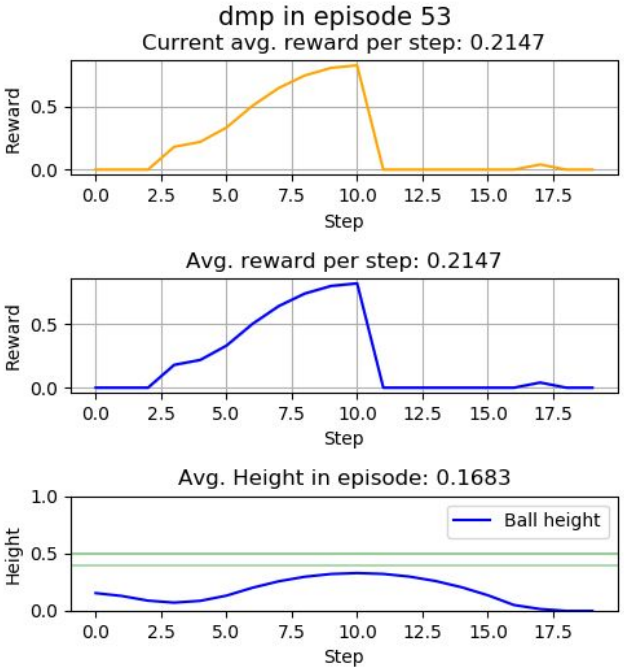
- **Usual approach**: Generate data with DMP to train DDPG
- **Our approach:** combine the DMP and DDPG action for each step and just feed the DDPG portion to the replay buffer for learning

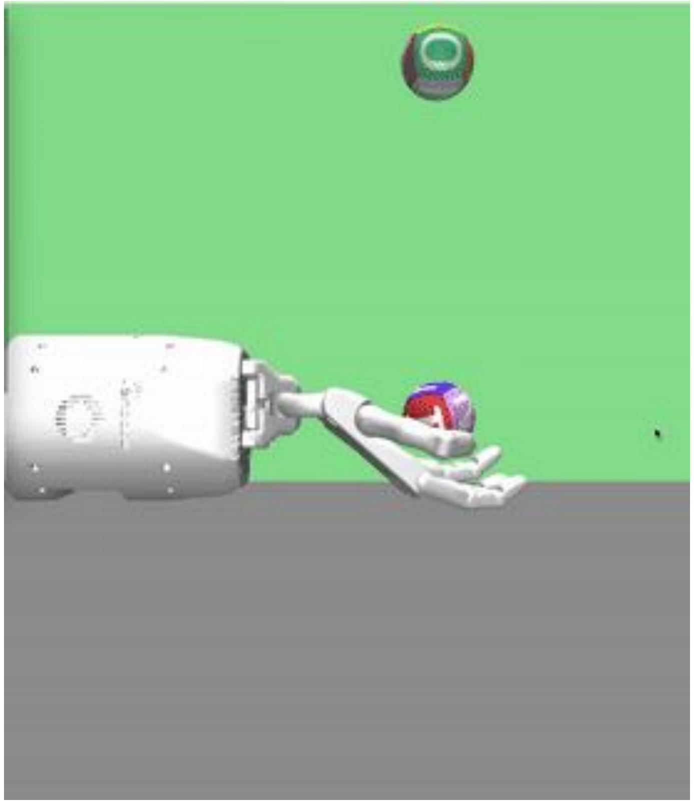
Evaluation: DDPG



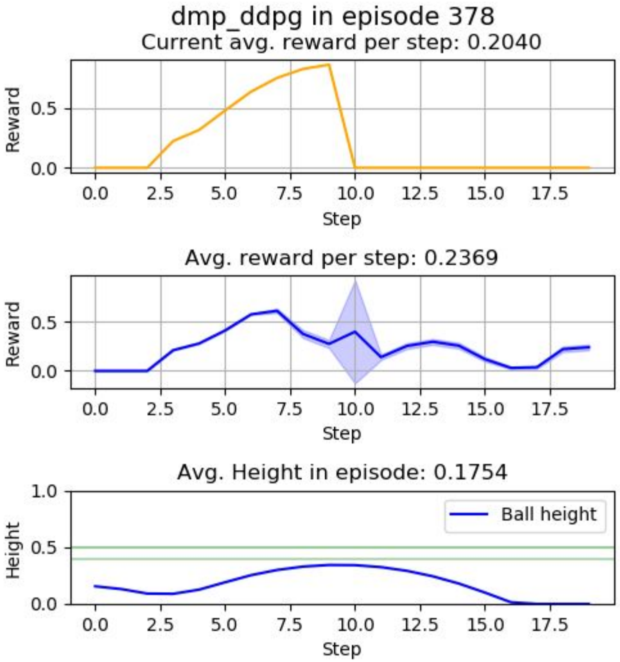


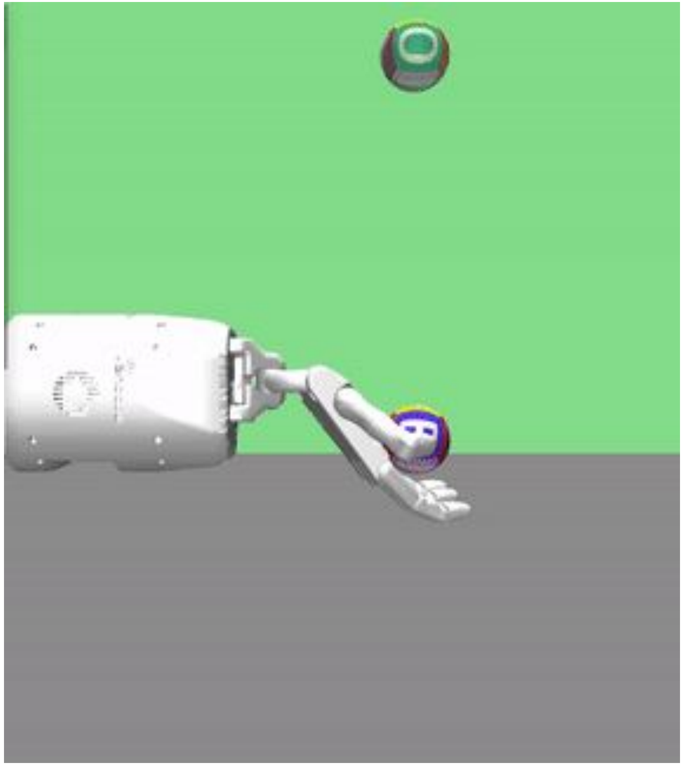
Evaluation: DMP





Evaluation: DDPG+DMP





Conclusion:

“We can teach an old robot a new trick!”

Successes:

- New environment
- Exciting insight using combined methods
- Learning about multiple challenges e.g. Sparse Reward Problem

Future Work:

- Further reward functions
- Hyper parameter tuning
- Further Algorithms e.g. Proximal Policy Optimization, Hindsight Experience Replay, etc.

Literatur & Quellen

- https://medium.com/datadriveninvestor/training-a-robotic-arm-to-do-human-like-tasks-using-rl-8d3106c87aaf
- https://blog.floydhub.com/robotic-arm-control-deep-reinforcement-learning/
- http://gazebosim.org/
- https://ai.googleblog.com/2019/01/soft-actor-critic-deep-reinforcement.html
- https://github.com/Unity-Technologies/ml-agents
- https://openai.com/blog/learning-dexterity/
- https://bair.berkeley.edu/blog/2018/08/31/dexterous-manip/
- https://gym.openai.com/envs/HandManipulateEgg-v0/
- https://arxiv.org/pdf/1701.08878.pdf
- https://github.com/pemami4911/deep-rl