Combining Trajectories and Policies

Aravind Rajeswaran & Kendall Lowrey

May 2, 2018

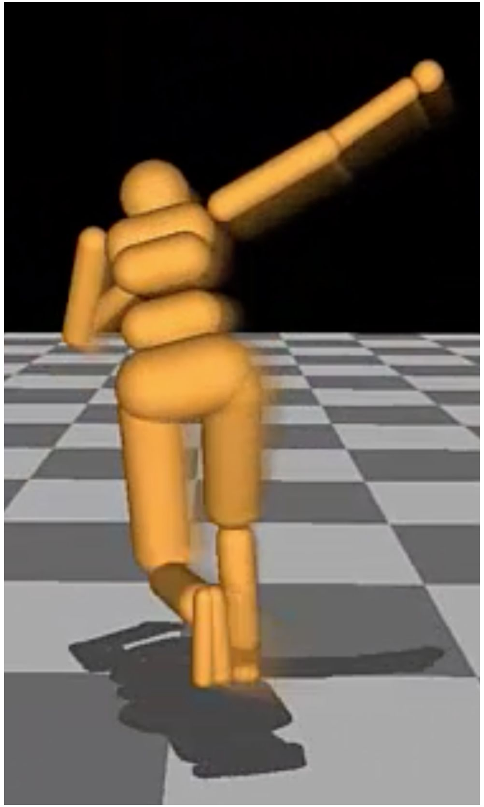
Last time...

We discussed how having access to a model can make for quick trajectory (aka local policy) learning.

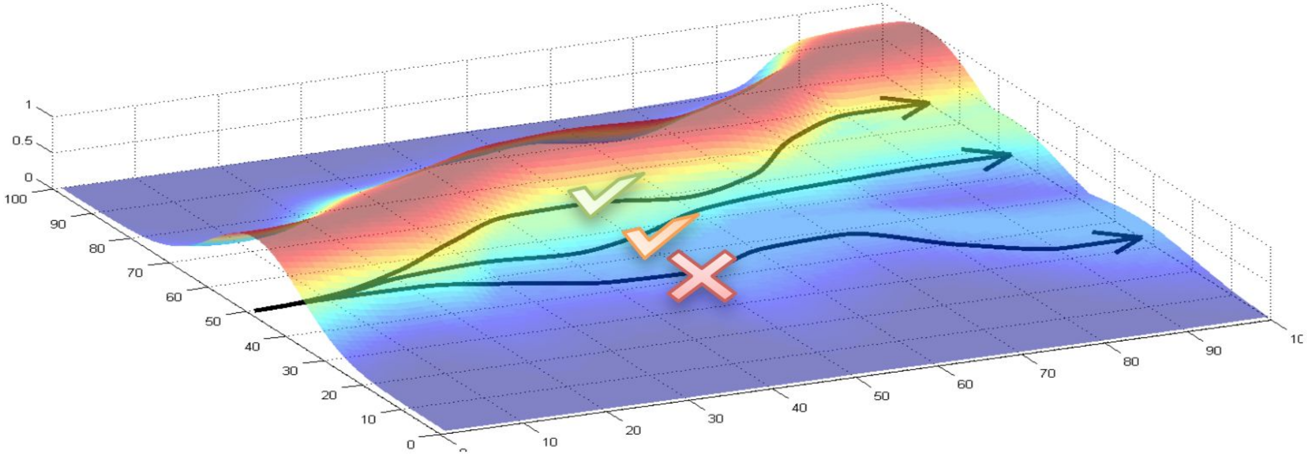
What a model is…

We walked through LQR as indirect trajectory optimization.

Discussed a method of direct optimization with soft constraints.



Trajectories	Policies
Exploits model to converge	No prior information (model) needed
Controller is very local	Policy can be “global”
Efficient but slow	“Robust” and fast to evaluate policy



Combining Trajectory Data with Policies

Behavior Cloning?

Distribution shift between trajectory data and policy

Backprop through model into Policy?

Policy now sensitive to model errors; long term coupled backprop problem

Imitation Learning (DAgger)?

No way to know when data augmentation is needed

Benefit of trajectories is we can always make more!

Problem

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$$

Have a model / Want to respect the system dynamics

$$\mathbf{u}_t = \pi_\theta(\mathbf{x}_t)$$

Want: A policy to tell us / agent what to do for a given state.

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T,\mathbf{x}_1,\ldots,\mathbf{x}_T} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t)$$

Want: a sequence of states and actions to result in low cost or high reward

Solution: Constrained Optimization

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T,\mathbf{x}_1,\ldots,\mathbf{x}_T,\theta} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t)$$

$$\text{s.t. } \mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$$

$$\text{s.t. } \mathbf{u}_t = \pi_\theta(\mathbf{x}_t)$$

$$\min_{\tau,\theta} c(\tau) \text{ s.t. } \mathbf{u}_t = \pi_\theta(\mathbf{x}_t)$$

Optimization with Constraints

Using Method of Lagrange Multipliers:

$$\min_{\mathbf{x}} f(\mathbf{x}) \text{ s.t. } C(\mathbf{x}) = 0 \qquad \mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda C(\mathbf{x})$$

We can instead do Dual Gradient Descent:

1. Find $\mathbf{x}^\star \leftarrow \arg\min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda)$
2. Compute $\frac{dg}{d\lambda} = \frac{d\mathcal{L}}{d\lambda}(\mathbf{x}^\star, \lambda)$ $\qquad g(\lambda) = \mathcal{L}(\mathbf{x}^\star(\lambda), \lambda)$
3. $\lambda \leftarrow \lambda + \alpha \frac{dg}{d\lambda}$

Guided Policy Search

$$\min_{\tau,\theta} c(\tau) \text{ s.t. } \mathbf{u}_t = \pi_\theta(\mathbf{x}_t)$$

$$\bar{\mathcal{L}}(\tau,\theta,\lambda) = c(\tau) + \sum_{t=1}^{T} \lambda_t(\pi_\theta(\mathbf{x}_t) - \mathbf{u}_t) + \sum_{t=1}^{T} \rho_t(\pi_\theta(\mathbf{x}_t) - \mathbf{u}_t)^2$$

1. Find $\tau \leftarrow \arg\min_\tau \bar{\mathcal{L}}(\tau,\theta,\lambda)$ (e.g. via iLQR)
2. Find $\theta \leftarrow \arg\min_\theta \bar{\mathcal{L}}(\tau,\theta,\lambda)$ (e.g. via SGD)
3. $\lambda \leftarrow \lambda + \alpha \frac{dg}{d\lambda}$

What about Gaussian Systems?

$$\min_{p,\theta} E_{\tau \sim p(\tau)}[c(\tau)] \text{ s.t. } p(\mathbf{u}_t|\mathbf{x}_t) = \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$$

$$\min_{p} \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\tilde{c}(\mathbf{x}_t,\mathbf{u}_t)] \text{ s.t. } D_{\mathrm{KL}}(p(\tau)\|\bar{p}(\tau)) \leq \epsilon$$

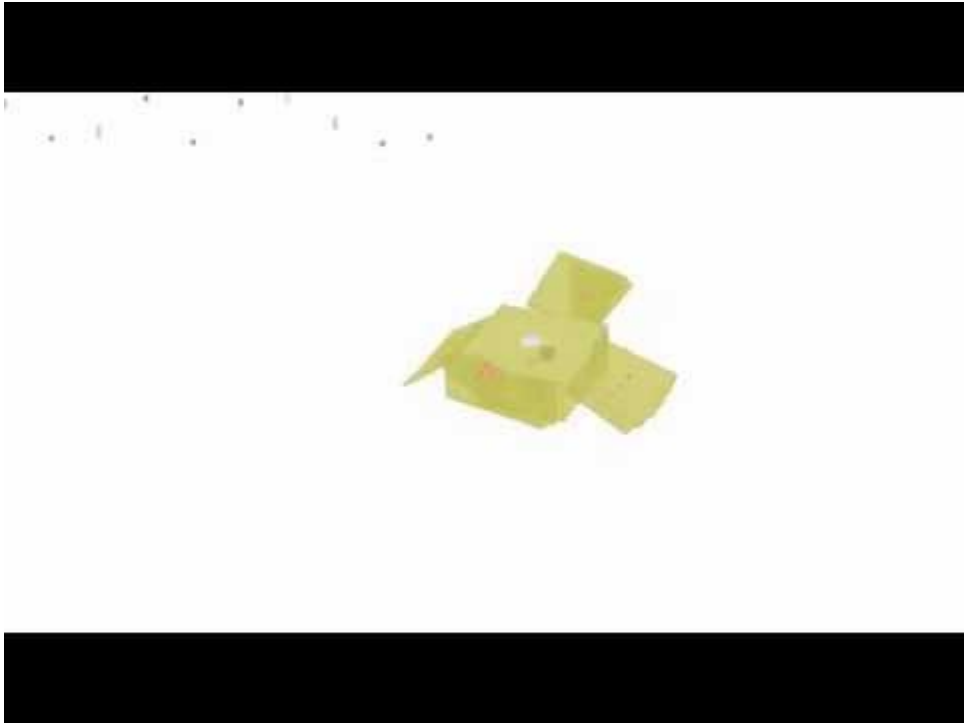
$$p(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t(\mathbf{x}_t - \hat{\mathbf{x}}_t) + \mathbf{k}_t + \hat{\mathbf{u}}_t, \Sigma_t) \qquad \bar{p}(\mathbf{u}_t|\mathbf{x}_t) = \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$$

$$p(\tau) = p(\mathbf{x}_1)\prod_{t=1}^{T} p(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t) \qquad \bar{p}(\tau) = \underline{p(\mathbf{x}_1)}\prod_{t=1}^{T} \bar{p}(\mathbf{u}_t|\mathbf{x}_t)\underline{p(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t)}$$

dynamics & initial state are the same!

Interactive Control of Diverse Complex Characters with Neural Networks

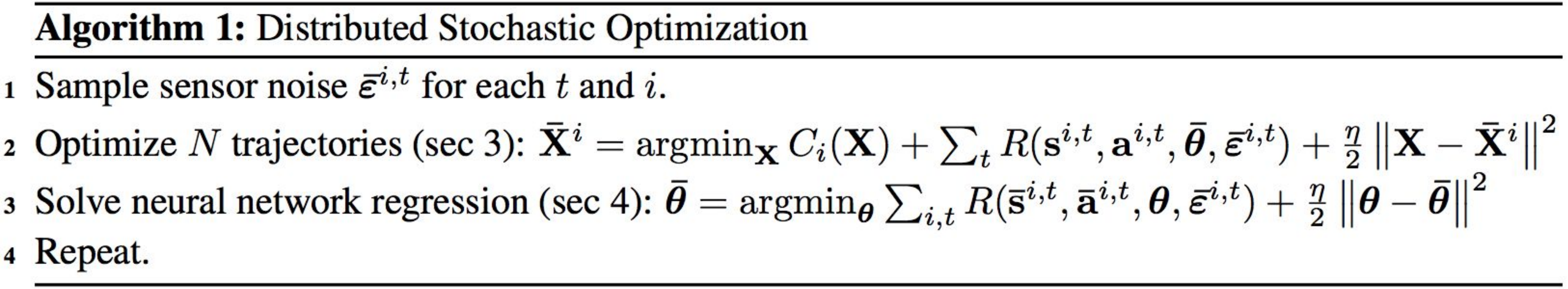
Igor Mordatch, Kendall Lowrey, Galen Andrew, Zoran Popovic, Emanuel Todorov



https://papers.nips.cc/paper/5764-interactive-control-of-diverse-complex-characters-with-neural-networks.pdf

Interactive Control of Diverse Complex Characters with Neural Networks

Igor Mordatch, Kendall Lowrey, Galen Andrew, Zoran Popovic, Emanuel Todorov



Algorithm 1: Distributed Stochastic Optimization

```
1 Sample sensor noise $\bar{\boldsymbol{\varepsilon}}^{i,t}$ for each $t$ and $i$.
2 Optimize $N$ trajectories (sec 3): $\bar{\mathbf{X}}^i = \mathrm{argmin}_{\mathbf{X}}\, C_i(\mathbf{X}) + \sum_t R(\mathbf{s}^{i,t}, \mathbf{a}^{i,t}, \bar{\boldsymbol{\theta}}, \bar{\boldsymbol{\varepsilon}}^{i,t}) + \frac{\eta}{2}\left\|\mathbf{X} - \bar{\mathbf{X}}^i\right\|^2$
3 Solve neural network regression (sec 4): $\bar{\boldsymbol{\theta}} = \mathrm{argmin}_{\boldsymbol{\theta}} \sum_{i,t} R(\bar{\mathbf{s}}^{i,t}, \bar{\mathbf{a}}^{i,t}, \boldsymbol{\theta}, \bar{\boldsymbol{\varepsilon}}^{i,t}) + \frac{\eta}{2}\left\|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\right\|^2$
4 Repeat.
```

Step 2 can be done in parallel across a cluster

Optimization is still soft constraints like in CIO, so no need for KL or other

Asynchronous: the soft constraints mean that our *N* trajectories isn't a fixed number

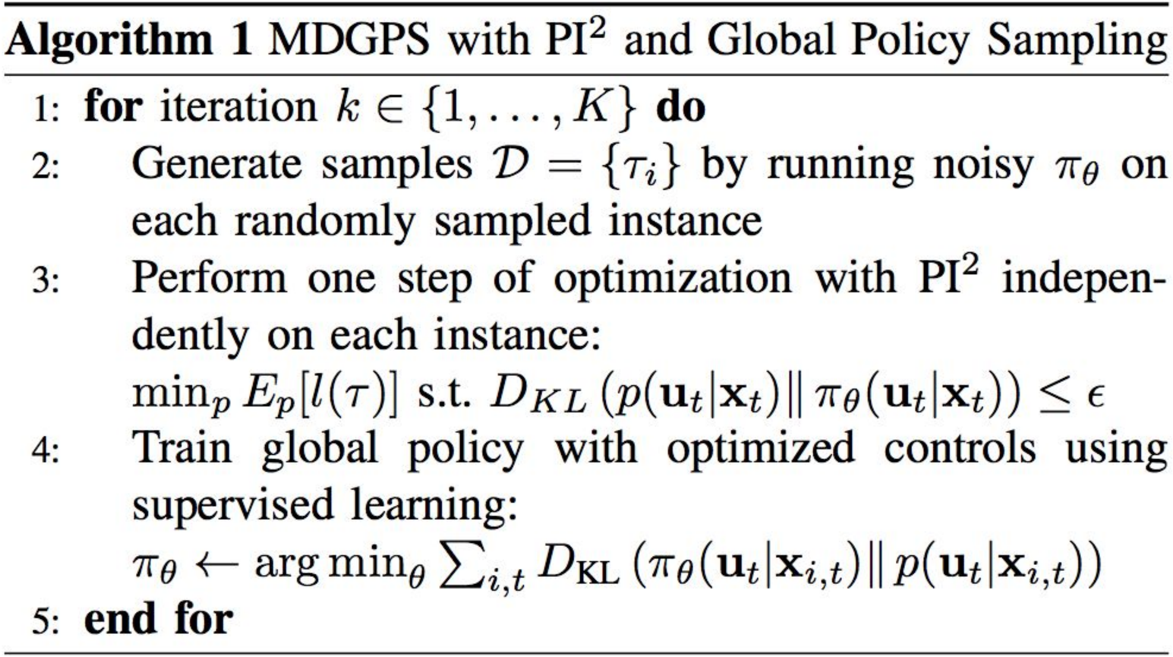
Path Integral Guided Policy Search

Yevgen Chebotar Mrinal Kalakrishnan Ali Yahya Adrian Li Stefan Schaal Sergey Levine

https://arxiv.org/pdf/1610.00529.pdf

Path Integral Guided Policy Search

Yevgen Chebotar Mrinal Kalakrishnan Ali Yahya Adrian Li Stefan Schaal Sergey Levine

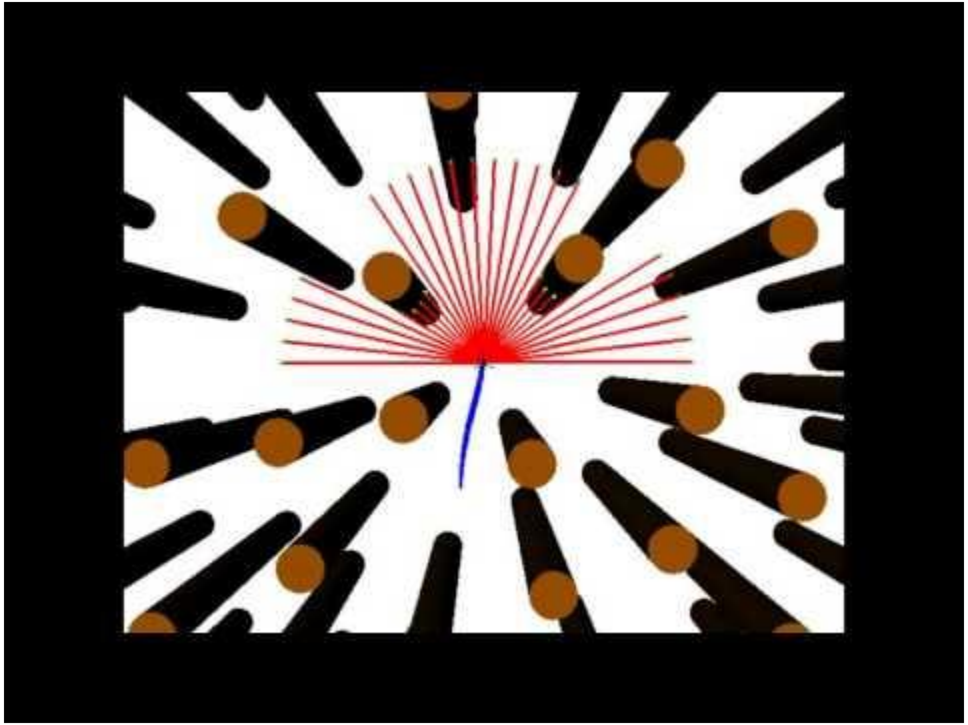


Algorithm 1 MDGPS with PI^2 and Global Policy Sampling

```
for iteration $k \in \{1, \dots, K\}$ do
    Generate samples $\mathcal{D} = \{\tau_i\}$ by running noisy $\pi_\theta$ on each randomly sampled instance
    Perform one step of optimization with PI$^2$ independently on each instance:
    $\min_p E_p[l(\tau)]$ s.t. $D_{KL}(p(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)) \leq \epsilon$
    Train global policy with optimized controls using supervised learning:
    $\pi_\theta \leftarrow \arg\min_\theta \sum_{i,t} D_{\mathrm{KL}}(\pi_\theta(\mathbf{u}_t|\mathbf{x}_{i,t}) \| p(\mathbf{u}_t|\mathbf{x}_{i,t}))$
end for
```

PLATO: Policy Learning with Adaptive Optimization

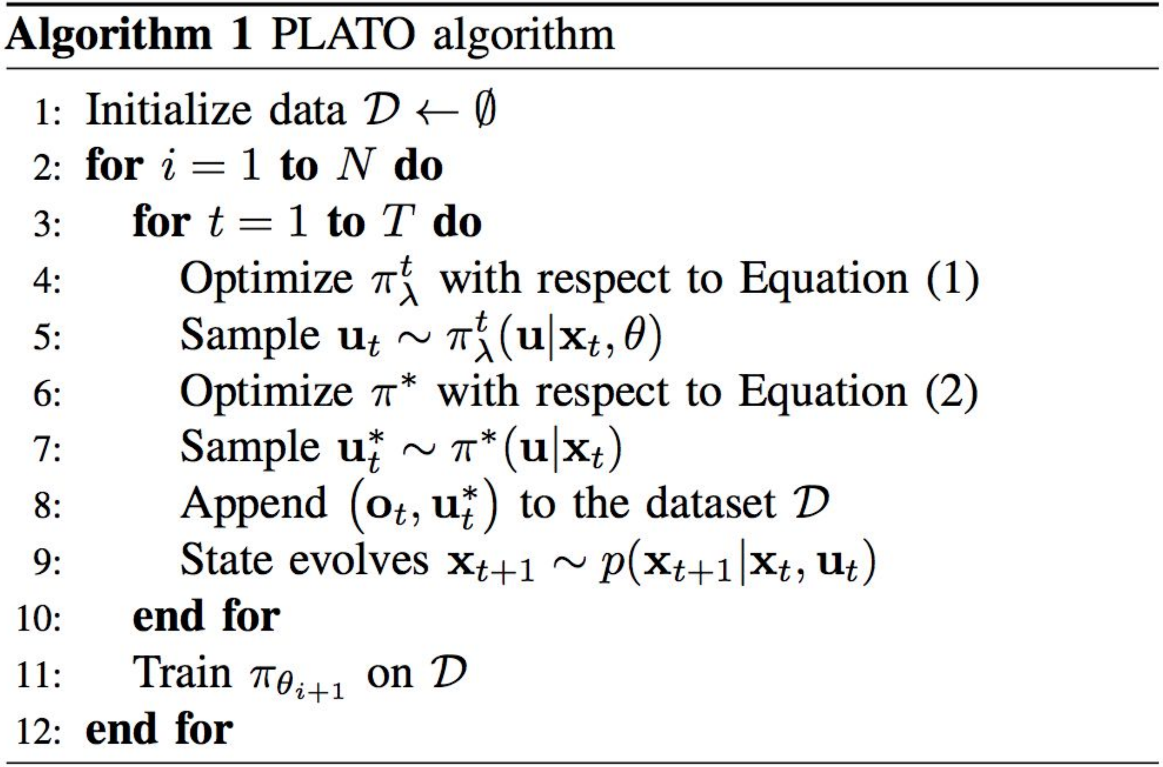
Gregory Kahn, Tianhao Zhang, Sergey Levine, Pieter Abbeel



https://arxiv.org/pdf/1603.00622.pdf

PLATO: Policy Learning with Adaptive Optimization

Gregory Kahn, Tianhao Zhang, Sergey Levine, Pieter Abbeel



Algorithm 1 PLATO algorithm

```
Initialize data $\mathcal{D} \leftarrow \emptyset$
for $i = 1$ to $N$ do
    for $t = 1$ to $T$ do
        Optimize $\pi_\lambda^t$ with respect to Equation (1)
        Sample $\mathbf{u}_t \sim \pi_\lambda^t(\mathbf{u}|\mathbf{x}_t, \theta)$
        Optimize $\pi^*$ with respect to Equation (2)
        Sample $\mathbf{u}_t^* \sim \pi^*(\mathbf{u}|\mathbf{x}_t)$
        Append $\left(\mathbf{o}_t, \mathbf{u}_t^*\right)$ to the dataset $\mathcal{D}$
        State evolves $\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$
    end for
    Train $\pi_{\theta_{i+1}}$ on $\mathcal{D}$
end for
```