End-to-End Training of Deep Visuomotor Policies

Sergey Levine, Chelsea Finn, Trevor Darrell, Pieter Abbeel
UC Berkeley

Presenter: Xixi Hu

Content

- Related work
- Method
- Experiments
- Discussions

Model-based Reinforcement Learning

What is “model-based RL”?

What is a “model”?

How does it differ from model-free RL?

Reference: https://docs.google.com/presentation/d/1f-DIrIvh44-jmTIKdKcue0Hx2RqQSw52t4k8HEdn5-c/edit#slide=id.g81ed7b35e4_0_2038

Model-free vs. model-based reinforcement learning

Collect data

$$\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^{T}$$

Model-free: learn policy directly from data

$$\mathcal{D} \rightarrow \pi$$

e.g. Q-learning, policy gradient

Model-based: learn model, then use it to learn or improve a policy

$$\mathcal{D} \rightarrow f \rightarrow \pi$$

Reference: https://docs.google.com/presentation/d/1f-DIrlvh44-jmTIKdKcue0Hx2RqQSw52t4k8HEdn5-c/edit#slide=id.g81ed7b35e4_0_2038

What is a model?

*Definition: a model is a representation that **explicitly** encodes knowledge about the structure of the environment and task.*

- A transition/dynamics model: $s_{t+1} = f_s(s_t, a_t)$
- A model of rewards: $r_{t+1} = f_r(s_t, a_t)$

Typically what is meant by the model in model-based RL

- An inverse transition/dynamics model: $a_t = f_s^{-1}(s_t, s_{t+1})$
- A model of distance: $d_{ij} = f_d(s_i, s_j)$
- A model of future returns: $G_t = Q(s_t, a_t)$ or $G_t = V(s_t)$

Reference: https://docs.google.com/presentation/d/1f-DIrIvh44-jmTIKdKcue0Hx2RqQSw52t4k8HEdn5-c/edit#slide=id.g81ed7b35e4_0_2038

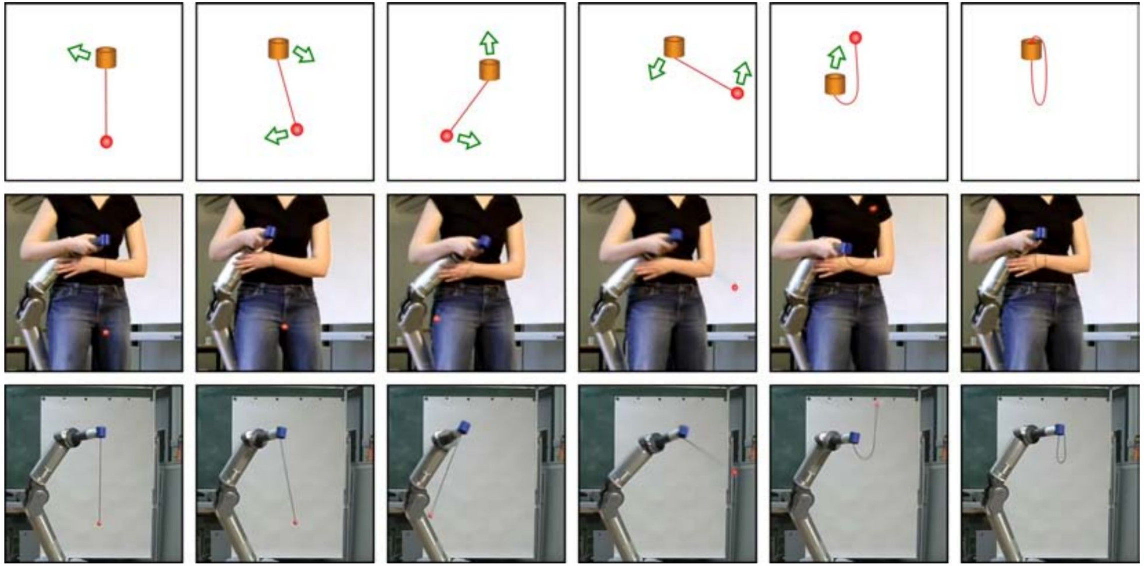
Learning Perception and Control Policy Separately

- **Separated vision pipeline and robot control pipeline**

Learning Perception and Control Policy Separately

- **Separated vision pipeline and robot control pipeline**

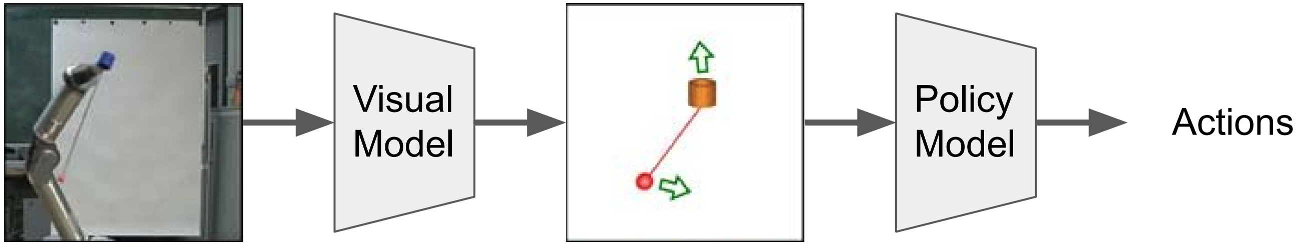
Example: Learning "Ball-in-the-Cup"



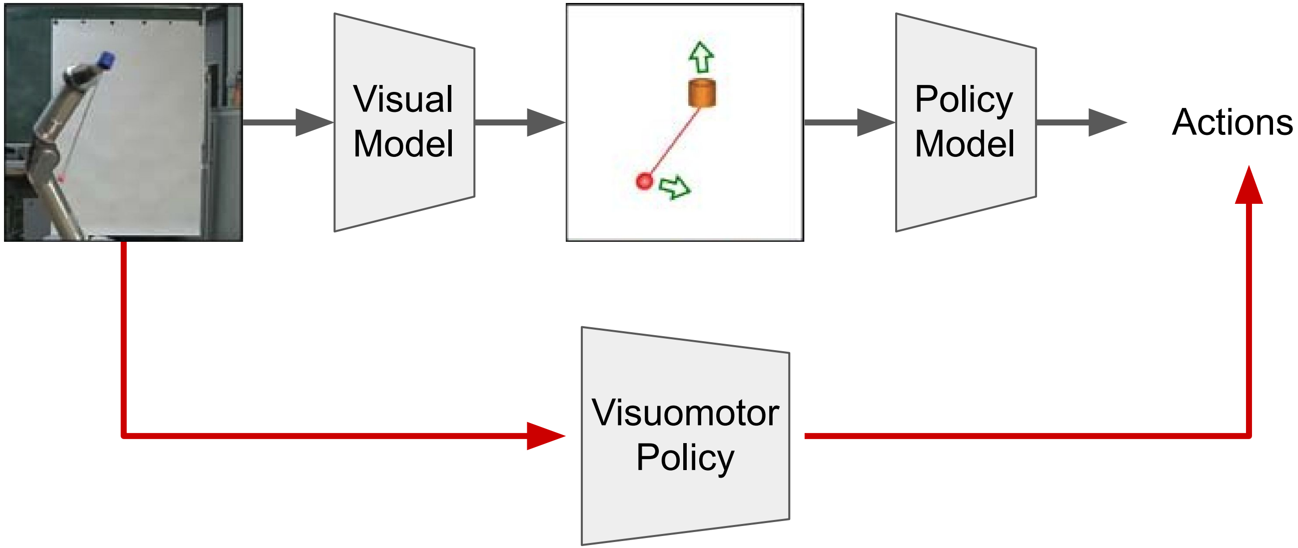
A stereo vision system was used to track the position of the ball. This ball position was used for determining the reward.

[1] Deisenroth, Marc Peter, Gerhard Neumann, and Jan Peters. "A survey on policy search for robotics." Foundations and Trends® in Robotics 2.1–2 (2013): 1-142.
[2] J. Kober and J. Peters. "Policy Search for Motor Primitives in Robotics." Machine Learning, pages 1–33, 2010.

Learning Perception and Control Policy Separately



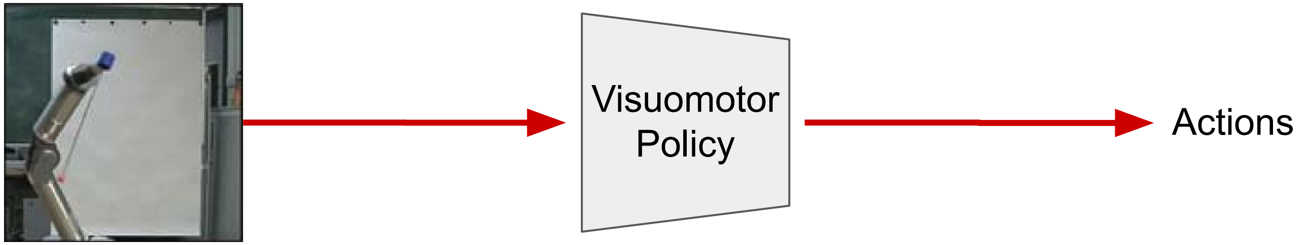
Learning Perception and Control Policy Jointly



Benefit:

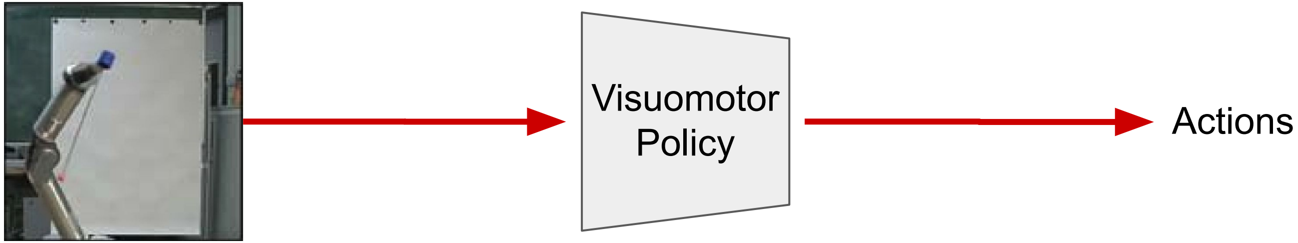
- Avoid hand-crafted design of visual perception model
- Perception gets better with policy training

Learning Perception and Control Policy Jointly



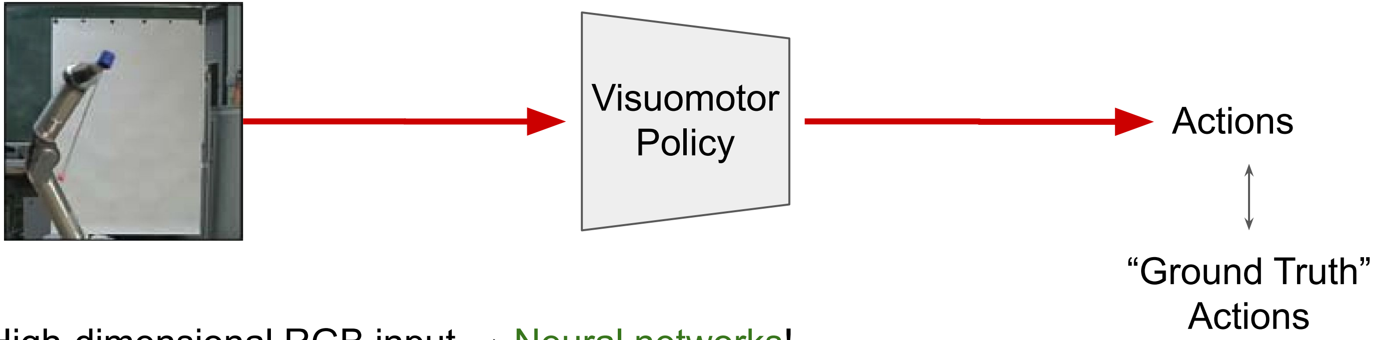
- High-dimensional RGB input → Deep neural networks!

Learning Perception and Control Policy Jointly



- High-dimensional RGB input → Deep neural networks!
- But …
  - Needs a lot of training data
  - Needs supervision of “ground-truth” actions

Learning Perception and Control Policy Jointly



- High-dimensional RGB input → Neural networks!
- But …
  - Needs a lot of training data
  - Needs supervision of "ground-truth" actions
- Solution: Generate "ground-truth actions" with trajectory optimization method

Related work

- **Application of deep learning on robotics control**
  - **Backpropagation: non-differentiable and instable**
  - **Not sample-efficient (unrealistic in real-world scene)**

Learning Perception and Control Policy Jointly

- What is trajectory optimization method?

$$\min_{u_1,\dots,u_t} \sum_{t=1}^{T} c(x_t, u_t) \text{ s.t. } x_t = f(x_{t-1}, u_{t-1})$$

u_t is the action at time step t

x_t is the state at time step t

f is the transition function

c is the cost function (negative reward of RL problem)

joint angles, end-effector pose, object positions, and their velocities; dimensionality: 14 to 32

- The controller learns a sequence of actions (trajectory) given fully observed state
- But it cannot generalize!
- In contrast, a policy can generalize better

Credit: https://michaelrzhang.github.io/model-based-rl

Learning Perception and Control Policy Jointly

- Trajectory Optimization Method → Guided Policy Search

$$\min_{u_1,\ldots,u_t} \sum_{t=1}^{T} c(x_t, u_t) \text{ s.t. } x_t = f(x_{t-1}, u_{t-1}) \longrightarrow \min_{\tau} c(\tau)$$

a sequence of actions (trajectory)

u_t is the action at time step t

x_t is the state at time step t

f is the transition function

c is the cost function (negative reward of RL problem)

Learning Perception and Control Policy Jointly

- Trajectory Optimization Method → Guided Policy Search

$$\min_{u_1,\ldots,u_t} \sum_{t=1}^{T} c(x_t, u_t) \text{ s.t. } x_t = f(x_{t-1}, u_{t-1})$$

u_t is the action at time step t
x_t is the state at time step t
f is the transition function
c is the cost function (negative reward of RL problem)

$$\min_{\tau} c(\tau)$$

$$\min_{\tau,\theta} c(\tau) \text{ s.t. } u_t = \pi_\theta(x_t)$$

Action generated by trajectory optimization method

Action generated by learned policy (visuomotor model)

Credit: https://michaelrzhang.github.io/model-based-rl

Learning Perception and Control Policy Jointly

- Trajectory Optimization Method → Guided Policy Search

$$\min_{u_1,\ldots,u_t} \sum_{t=1}^{T} c(x_t, u_t) \text{ s.t. } x_t = f(x_{t-1}, u_{t-1})$$

u_t is the action at time step t
x_t is the state at time step t
f is the transition function
c is the cost function (negative reward of RL problem)

$$\min_{\tau} c(\tau)$$

$$\min_{\tau,\theta} c(\tau) \text{ s.t. } u_t = \pi_\theta(x_t)$$

↓ Lagrangian

$$\mathcal{L}(\tau, \theta, \lambda) = c(\tau) + \sum_{t=1}^{T} \lambda_t(\pi_\theta(x_t) - u_t)$$

Credit: https://michaelrzhang.github.io/model-based-rl

Learning Perception and Control Policy Jointly

- Guided Policy Search - Optimization

$$\mathcal{L}(\tau, \theta, \lambda) = c(\tau) + \sum_{t=1}^{T} \lambda_t(\pi_\theta(x_t) - u_t)$$

Standard optimization problem
Solve it using ADMM

1. Start with some initial choice of λ (by λ, we include λ_t corresponding to each time step)
2. Assign $\tau \leftarrow \arg\min_\tau \mathcal{L}(\tau, \theta, \lambda)$.
3. Assign $\theta \leftarrow \arg\min_\theta \mathcal{L}(\tau, \theta, \lambda)$.
4. Compute $\frac{dg}{d\lambda} = \frac{d\mathcal{L}}{d\lambda}(\tau, \theta, \lambda)$. Take a gradient step $\lambda \leftarrow \lambda + \alpha \frac{dg}{d\lambda}$
5. Repeat steps 2-4.

Credit: https://michaelrzhang.github.io/model-based-rl

Learning Perception and Control Policy Jointly

- Guided Policy Search - Optimization

$$\mathcal{L}(\tau, \theta, \lambda) = c(\tau) + \sum_{t=1}^{T} \lambda_t(\pi_\theta(x_t) - u_t)$$

Standard optimization problem
Solve it using ADMM

1. Start with some initial choice of λ (by λ, we include λ_t corresponding to each time step)
2. Assign $\tau \leftarrow \arg\min_\tau \mathcal{L}(\tau, \theta, \lambda)$.
3. Assign $\theta \leftarrow \arg\min_\theta \mathcal{L}(\tau, \theta, \lambda)$.
4. Compute $\frac{dg}{d\lambda} = \frac{d\mathcal{L}}{d\lambda}(\tau, \theta, \lambda)$. Take a gradient step $\lambda \leftarrow \lambda + \alpha \frac{dg}{d\lambda}$
5. Repeat steps 2-4.

Use some trajectory optimization methods to solve it (→ step 2)

Credit: https://michaelrzhang.github.io/model-based-rl

Learning Perception and Control Policy Jointly

- Guided Policy Search - Optimization

$$\mathcal{L}(\tau, \theta, \lambda) = c(\tau) + \sum_{t=1}^{T} \lambda_t(\pi_\theta(x_t) - u_t)$$

Standard optimization problem
Solve it using ADMM

1. Start with some initial choice of λ (by λ, we include λ_t corresponding to each time step)
2. Assign $\tau \leftarrow \arg\min_\tau \mathcal{L}(\tau, \theta, \lambda)$. → Use some trajectory optimization methods to solve it
3. Assign $\theta \leftarrow \arg\min_\theta \mathcal{L}(\tau, \theta, \lambda)$. → Supervised learning
4. Compute $\frac{dg}{d\lambda} = \frac{d\mathcal{L}}{d\lambda}(\tau, \theta, \lambda)$. Take a gradient step $\lambda \leftarrow \lambda + \alpha \frac{dg}{d\lambda}$
5. Repeat steps 2-4.

Credit: https://michaelrzhang.github.io/model-based-rl

Learning Perception and Control Policy Jointly

- Guided Policy Search

$$\mathcal{L}(\tau, \theta, \lambda) = c(\tau) + \sum_{t=1}^{T} \lambda_t(\pi_\theta(x_t) - u_t)$$

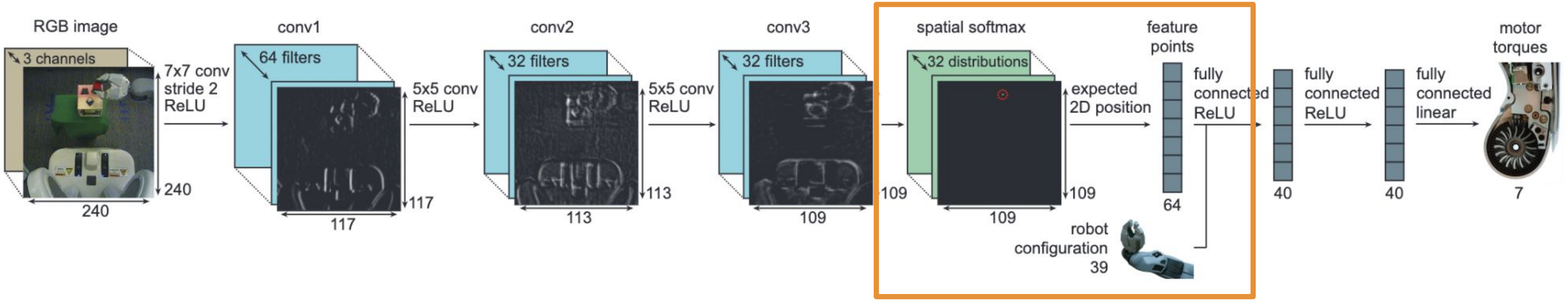
1. Start with some initial choice of λ (by λ, we include λ_t corresponding to each time step)
2. Assign $\tau \leftarrow \arg\min_\tau \mathcal{L}(\tau, \theta, \lambda)$.
3. Assign $\theta \leftarrow \arg\min_\theta \mathcal{L}(\tau, \theta, \lambda)$.
4. Compute $\frac{dg}{d\lambda} = \frac{d\mathcal{L}}{d\lambda}(\tau, \theta, \lambda)$. Take a gradient step $\lambda \leftarrow \lambda + \alpha \frac{dg}{d\lambda}$
5. Repeat steps 2-4.

Recap:

- Each trajectory-centric teacher only needs to solve the task from a single initial state → make the problem easier
- The policy is trained with supervised learning → good generalization
- Iterative adaptation of teacher trajectories & final policy → the teacher does not take actions that the final policy cannot reproduce

Credit: https://michaelrzhang.github.io/model-based-rl

Visuomotor Policy Architecture



$$s_{cij} = e^{a_{cij}} / \sum_{i'j'} e^{a_{ci'j'}}$$

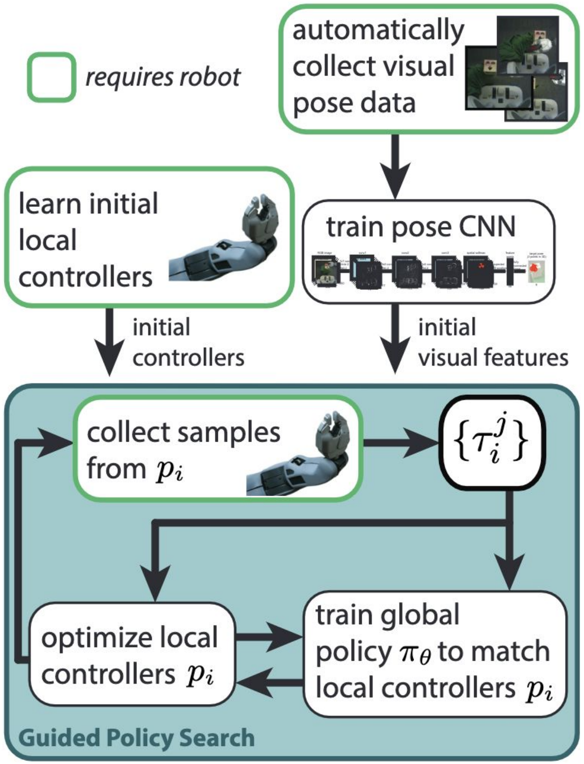
$$f_{cx} = \sum_{ij} s_{cij} x_{ij}$$

$$f_{cy} = \sum_{ij} s_{cij} y_{ij}$$

- Spatial softmax → soft version of max pooling
- Get the feature points
- Learns the spatial information better

Training procedural

- Pretraining convolutional layers
- Pretraining local controller
- End-to-end guided policy search



Method

- **Algorithm**

Target:

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

$$\ell(\tau) = \sum_{t=1}^{T} \ell(\mathbf{x}_t, \mathbf{u}_t)$$

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

Method

$$\ell(\tau) = \sum_{t=1}^{T} \ell(\mathbf{x}_t, \mathbf{u}_t)$$

- **Algorithm**

$$\mathcal{L}_\theta(\theta, p) = \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t)] + E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] - E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] + \nu_t \phi_t^\theta(\theta, p)$$

$$\mathcal{L}_p(p, \theta) = \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t)] + E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] - E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] + \nu_t \phi_t^p(\theta, p),$$

$$\phi_t^p(p, \theta) = E_{p(\mathbf{x}_t)}[D_{\mathrm{KL}}(p(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{x}_t))]$$

$$\phi_t^\theta(\theta, p) = E_{p(\mathbf{x}_t)}[D_{\mathrm{KL}}(\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)) \| p(\mathbf{u}_t|\mathbf{x}_t)].$$

$$\theta \leftarrow \arg\min_\theta \sum_{t=1}^{T} E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] + \nu_t \phi_t^\theta(\theta, p)$$

$$p \leftarrow \arg\min_p \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t) - \lambda_{\mathbf{x}_t,\mathbf{u}_t}] + \nu_t \phi_t^p(p, \theta)$$

$$\lambda_{\mathbf{x}_t,\mathbf{u}_t} \leftarrow \lambda_{\mathbf{x}_t,\mathbf{u}_t} + \alpha\nu_t(\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t) - p(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)).$$

ADMM

$$\min_{\mathbf{x}\in\mathbb{R}^n, \mathbf{z}\in\mathbb{R}^m} \quad f(\mathbf{x}) + g(\mathbf{z})$$
$$\text{s.t.} \quad \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}$$

$$L\rho(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{z}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2$$

minimization step for $\mathbf{x}$

$$\mathbf{x}^{k+1} \triangleq \arg\min_{\mathbf{x}\in\mathbb{R}^n} \quad L_\rho(\mathbf{x}, \mathbf{z}^k, \mathbf{y}^k)$$

minimization step for $\mathbf{z}$

$$\mathbf{z}^{k+1} \triangleq \arg\min_{\mathbf{z}\in\mathbb{R}^m} \quad L_\rho(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{y}^k)$$

for dual variable update

$$\mathbf{y}^{k+1} \triangleq \mathbf{y}^k + \rho(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^{k+1} - \mathbf{b})$$

https://abhishekaich27.github.io/data/WriteUps/ADMM_Review.pdf

Method

- **Algorithm**

$$\theta \leftarrow \arg\min_{\theta} \sum_{t=1}^{T} E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] + \nu_t \phi_t^\theta(\theta, p)$$

$$p \leftarrow \arg\min_{p} \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t) - \lambda_{\mathbf{x}_t,\mathbf{u}_t}] + \nu_t \phi_t^p(p, \theta)$$

$$\lambda_{\mathbf{x}_t,\mathbf{u}_t} \leftarrow \lambda_{\mathbf{x}_t,\mathbf{u}_t} + \alpha\nu_t(\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t) - p(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)).$$

$$\theta \leftarrow \arg\min_{\theta} \sum_{t=1}^{T} E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}[\mathbf{u}_t^{\mathrm{T}} \lambda_{\mu t}] + \nu_t \phi_t^\theta(\theta, p)$$

$$p \leftarrow \arg\min_{p} \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t) - \mathbf{u}_t^{\mathrm{T}} \lambda_{\mu t}] + \nu_t \phi_t^p(p, \theta)$$

$$\lambda_{\mu t} \leftarrow \lambda_{\mu t} + \alpha\nu_t(E_{\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)}[\mathbf{u}_t] - E_{p(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)}[\mathbf{u}_t]),$$

Method

$$\mathcal{L}_p(p,\theta) = \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t,\mathbf{u}_t)] + E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] - E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] + \nu_t \phi_t^p(\theta,p),$$

$$p \leftarrow \arg\min_p \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t,\mathbf{u}_t) - \mathbf{u}_t^{\mathrm{T}} \lambda_{\mu t}] + \nu_t \phi_t^p(p,\theta)$$

- **Trajectory optimization under unknown dynamics**

$$p(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \mathbf{C}_t) \qquad p(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t) = \mathcal{N}(f_{\mathbf{x}t}\mathbf{x}_t + f_{\mathbf{u}t}\mathbf{u}_t + f_{ct}, \mathbf{F}_t).$$

$$\min_{p(\tau)\in\mathcal{N}(\tau)} \mathcal{L}_p(p,\theta) \text{ s.t. } D_{\mathrm{KL}}(p(\tau)\|\hat{p}(\tau)) \le \epsilon.$$

Method

$$\mathcal{L}_\theta(\theta, p) = \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t,\mathbf{u}_t)] + E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] - E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] + \nu_t \phi_t^\theta(\theta, p)$$

$$\theta \leftarrow \arg\min_\theta \sum_{t=1}^{T} E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}[\mathbf{u}_t^{\mathrm{T}}\lambda_{\mu t}] + \nu_t \phi_t^\theta(\theta, p)$$

- **Supervised policy optimization**

$$\mathcal{L}_\theta(\theta, p) = \frac{1}{2N}\sum_{i=1}^{N}\sum_{t=1}^{T} E_{p_i(\mathbf{x}_t,\mathbf{o}_t)}\Big[\mathrm{tr}[\mathbf{C}_{ti}^{-1}\Sigma^\pi(\mathbf{o}_t)] - \log|\Sigma^\pi(\mathbf{o}_t)| + (\mu^\pi(\mathbf{o}_t) - \mu_{ti}^p(\mathbf{x}_t))\mathbf{C}_{ti}^{-1}(\mu^\pi(\mathbf{o}_t) - \mu_{ti}^p(\mathbf{x}_t)) + 2\lambda_{\mu t}^{\mathrm{T}}\mu^\pi(\mathbf{o}_t)\Big],$$

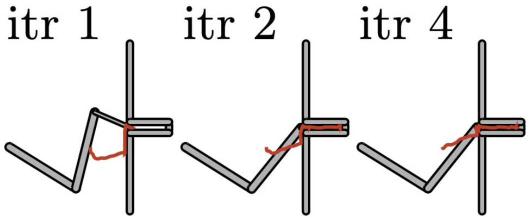
$$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t) = \mathcal{N}(\mu^\pi(\mathbf{o}_t), \Sigma^\pi(\mathbf{o}_t)).$$

Experiments

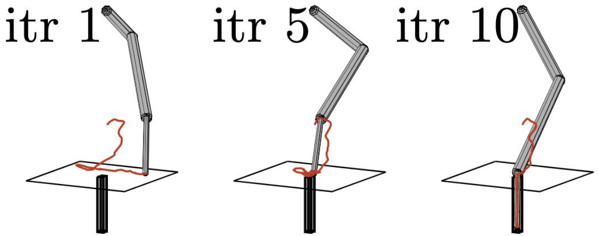
- How does the guided policy search algorithm compare to other policy search methods for training complex, high-dimensional policies, such as neural networks?
- Does our trajectory optimization algorithm work on a real robotic platform with unknown dynamics, for a range of different tasks?
- How does our spatial softmax architecture compare to other, more standard convolutional neural network architectures?
- Does training the perception and control systems in a visuomotor policy jointly end-to-end provide better performance than training each component separately?

Experiments

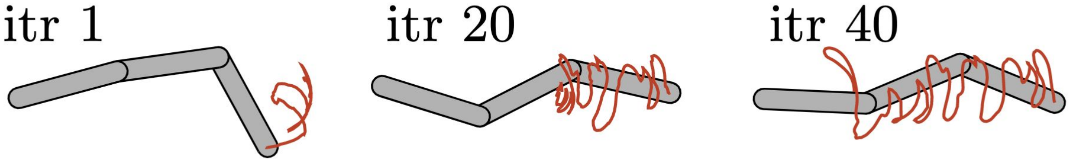
- How does the guided policy search algorithm compare to other policy search methods for training complex, high-dimensional policies, such as neural networks?



2D insertion

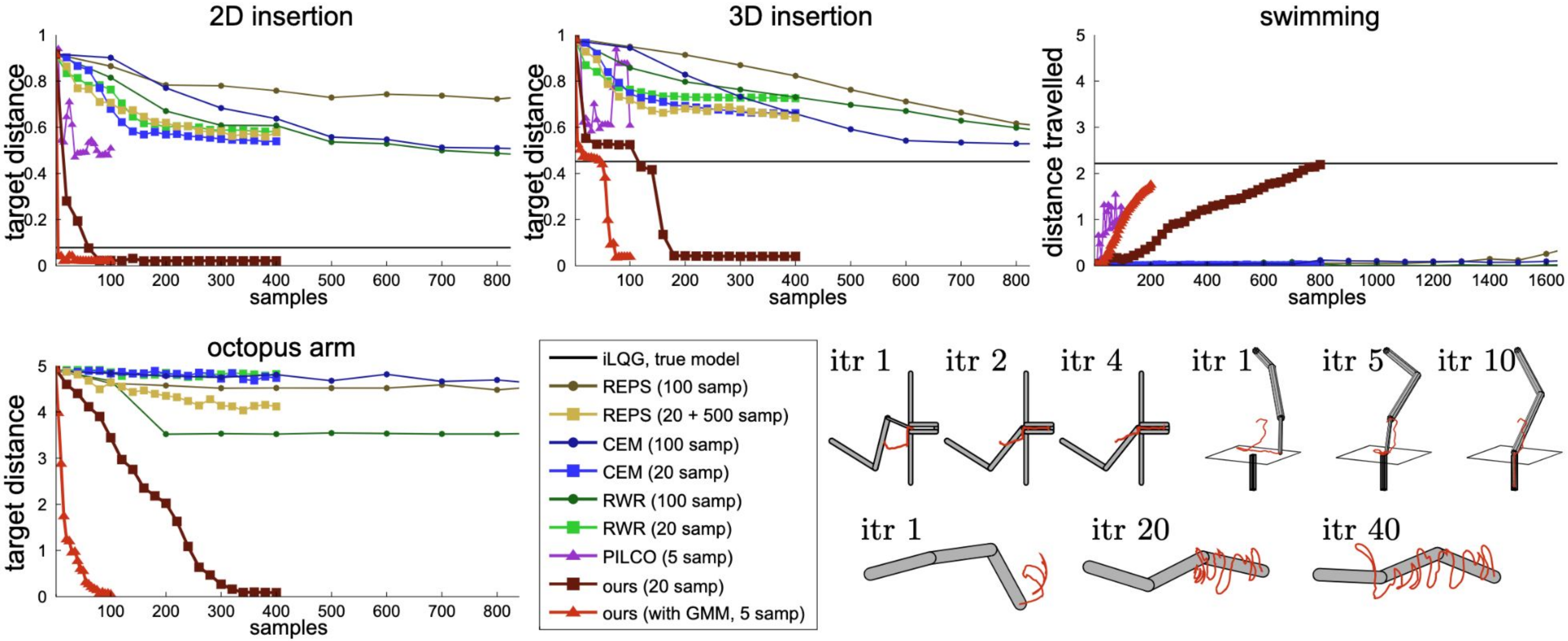


3D insertion

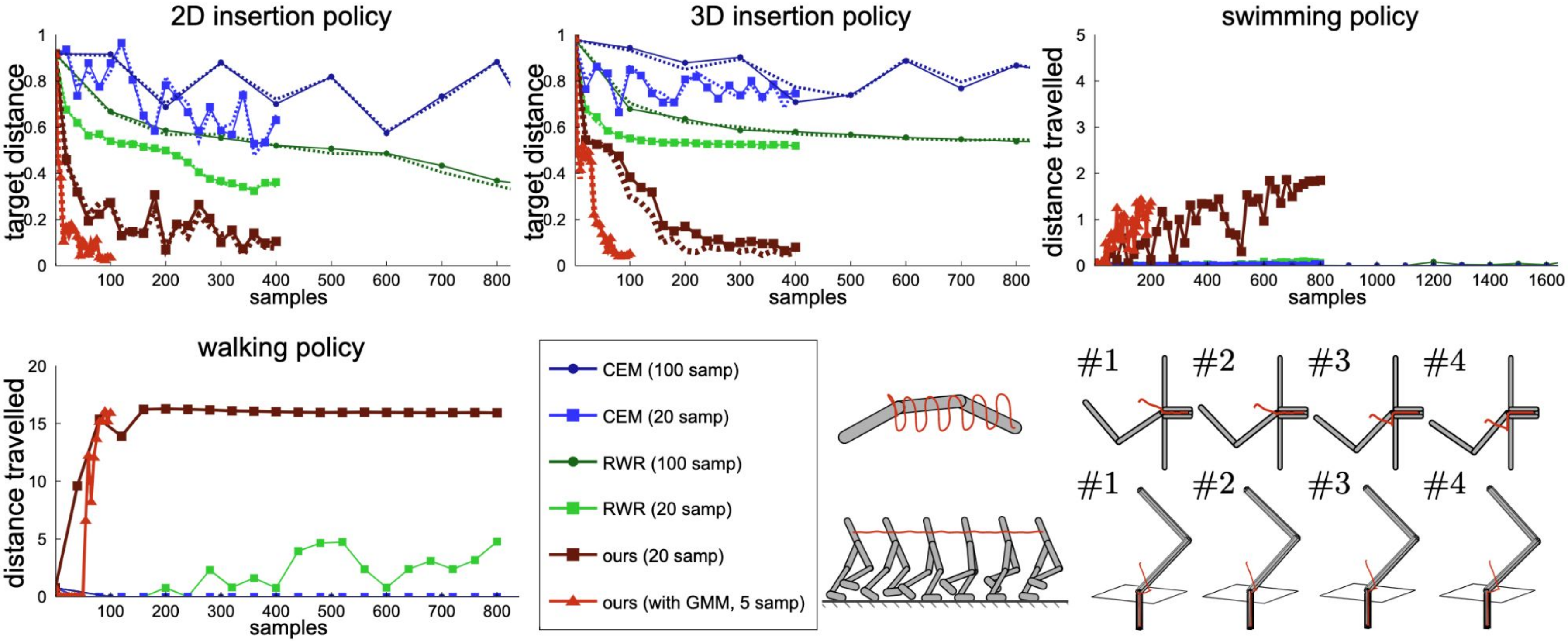


Swimming

Experiments

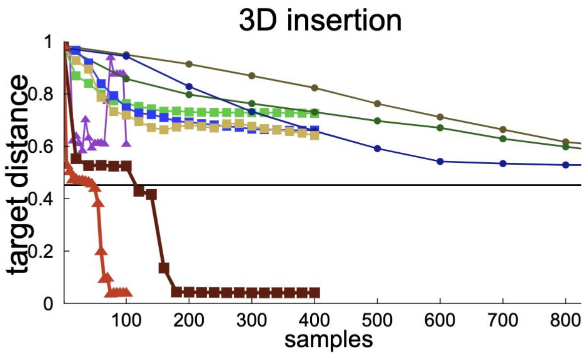


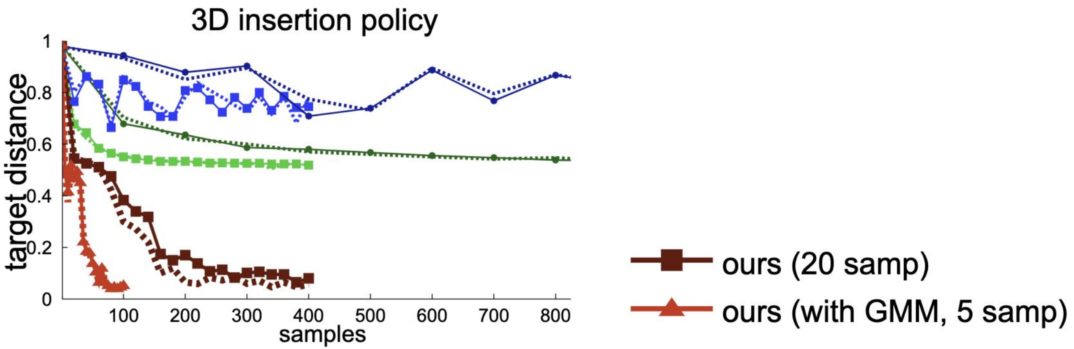
Experiments



Experiments

- How does the guided policy search algorithm compare to other policy search methods for training complex, high-dimensional policies, such as neural networks?
    - Compare with methods that do not use vision, but use system states
    - GPS is more sample efficient
    - Better generalization in testing

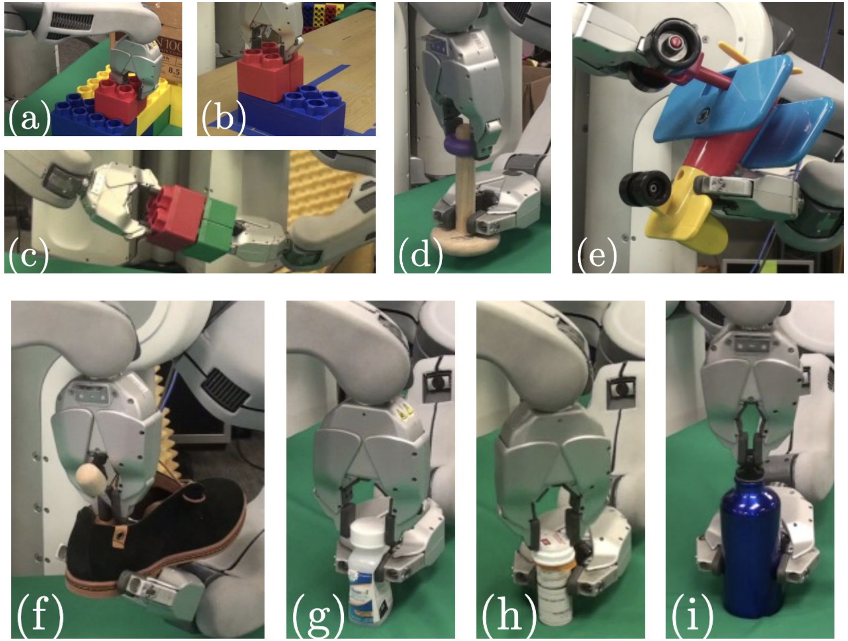




Experiments

Experiments

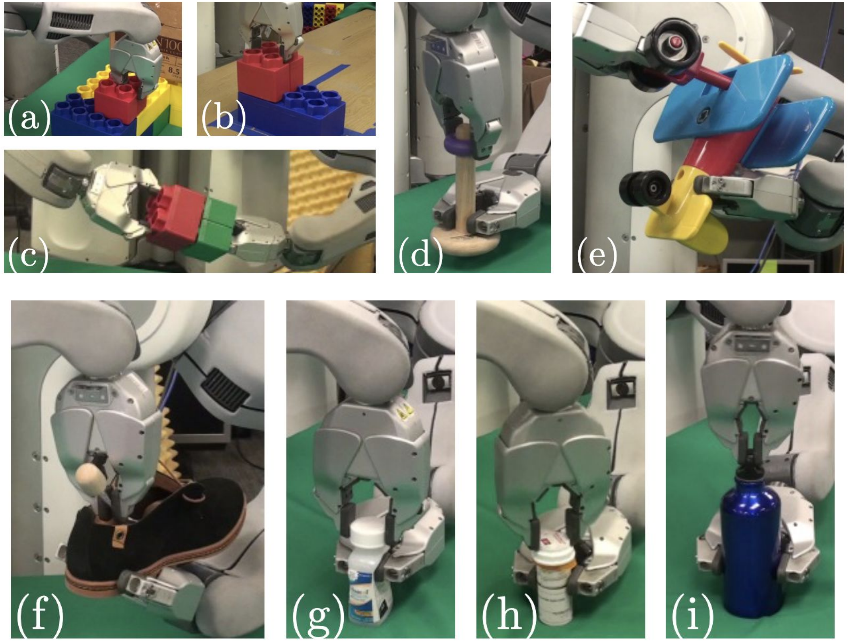
- Does our trajectory optimization algorithm work on a real robotic platform with unknown dynamics, for a range of different tasks?



  - Sample efficient (20-25 samples)
  - Robustness

Experiments

- Does our trajectory optimization algorithm work on a real robotic platform with unknown dynamics, for a range of different tasks?
  - Sample efficient (20-25 samples)
  - Robustness



Experiments

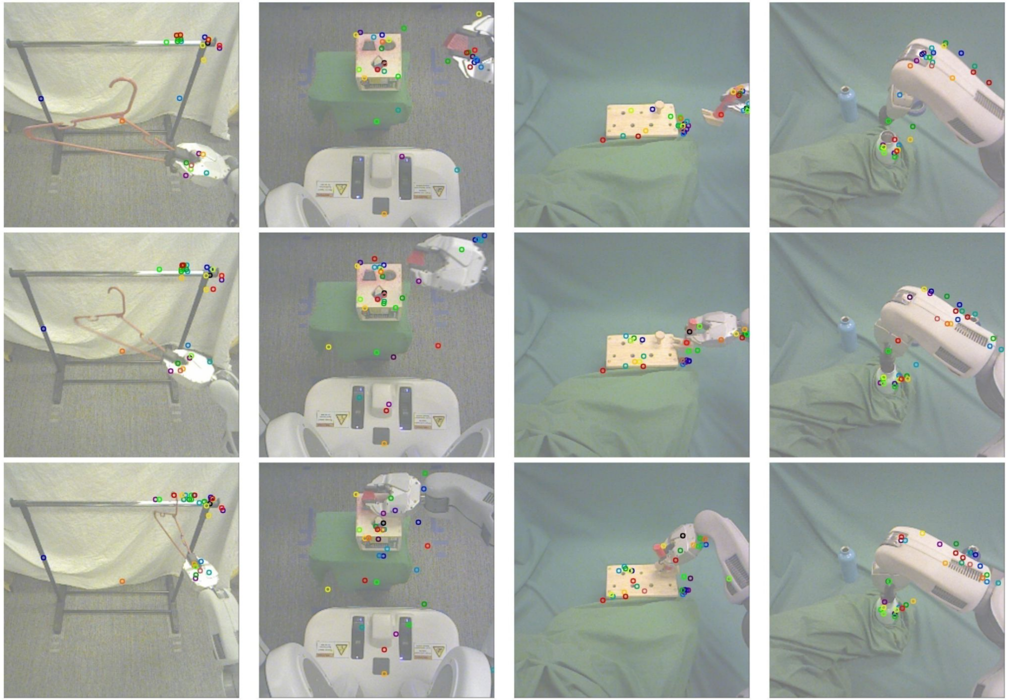
- How does our spatial softmax architecture compare to other, more standard convolutional neural network architectures?

network architecture	test error (cm)
softmax + feature points (**ours**)	$\mathbf{1.30 \pm 0.73}$
softmax + fully connected layer	2.59 ± 1.19
fully connected layer	4.75 ± 2.29
max-pooling + fully connected	3.71 ± 1.73

Performance on Object Pose Estimation Pretraining Task

Experiments

- Visualization of feature points



(a) hanger (b) cube (c) hammer (d) bottle

Experiments

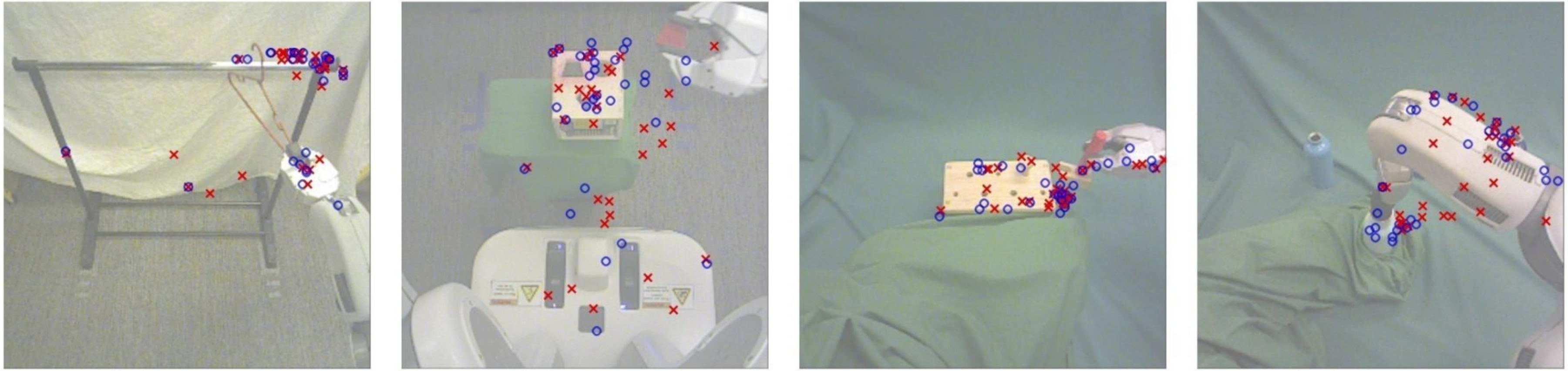
- Does training the perception and control systems in a visuomotor policy jointly end-to-end provide better performance than training each component separately?

Baselines:
train the vision layers in advance,
then train the policy

coat hanger	training (18)	spatial test (24)	visual test (18)
end-to-end	**100%**	**100%**	**100%**
pose features	88.9%	87.5%	83.3%
pose prediction	55.6%	58.3%	66.7%
shape cube	training (27)	spatial test (36)	visual test (40)
end-to-end	**96.3%**	**91.7%**	**87.5%**
pose features	70.4%	83.3%	40%
pose prediction	0%	0%	n/a
toy hammer	training (45)	spatial test (60)	visual test (60)
end-to-end	**91.1%**	**86.7%**	**78.3%**
pose features	62.2%	75.0%	53.3%
pose prediction	8.9%	18.3%	n/a
bottle cap	training (27)	spatial test (12)	visual test (40)
end-to-end	**88.9%**	**83.3%**	**62.5%**
pose features	55.6%	58.3%	27.5%

Success rates on training positions, on novel test positions, and in the presence of visual distractors. The number of trials per test is shown in parentheses.

Experiments



(a) hanger (b) cube (c) hammer (d) bottle

Discussions

- What are the drawbacks/limitations for this method?

Discussions

- Needs Full-state observations during training

Discussions

- Needs Full-state observations during training
- Dependent on hand-crafted rewards

Discussions

- Needs Full-state observations during training
- Dependent on hand-crafted rewards
- Relying on the ability of Trajectory Optimization Method (teachers) to discover good trajectories

Questions?