Covariance Matrix Adaptation for Direct Reinforcement Learning

Freek Stulp
Cognitive Robotics Group – ENSTA-ParisTech
FLOWERS Team – INRIA/Bordeaux

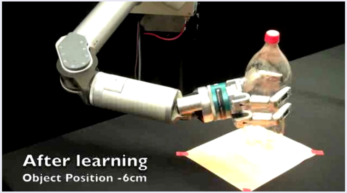
Olivier Sigaud
Institut des Systmes Intelligents et de Robotique – UPMC CNRS UMR 7222

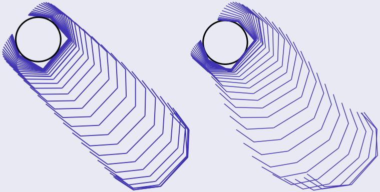
JFPDA 23.05.2012

Motivation

- PI^2 powerful algorithm for direct RL on robots

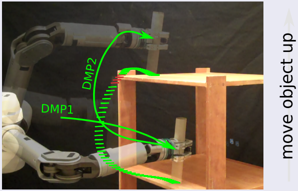
Grasping under Uncertainty

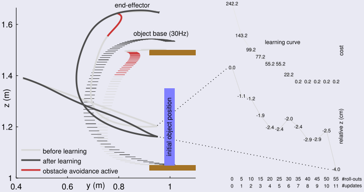




IROS'11, ICRA'11

Pick-and-Place Tasks





Humanoids'11

Motivation

- PI^2 powerful algorithm for direct RL on robots
  - Tuning exploration parameter in PI^2

Motivation

- PI^2 powerful algorithm for direct RL on robots
  - Tuning exploration parameter in PI^2
- Update rules for PI^2 and $\mathrm{CEM}/\mathrm{CMA\text{-}ES}$ almost identical
  - But CEM and $\mathrm{CMA\text{-}ES}$ additionaly tune exploration parameter automatically

Motivation

- PI^2 powerful algorithm for direct RL on robots
  - Tuning exploration parameter in PI^2
- Update rules for PI^2 and $\mathrm{CEM}/\mathrm{CMA\text{-}ES}$ almost identical
  - But CEM and $\mathrm{CMA\text{-}ES}$ additionaly tune exploration parameter automatically
- Goals
  - Analysis and comparison of $\mathsf{PI}^2/\mathrm{CEM}/\mathrm{CMA\text{-}ES}$
  - Novel algorithm PI^2-CMA
    - Essentially PI^2 with automatic exploration parameter tuning

Outline

$\boxed{\text{PI}^2}$

Outline

PI^2

CEM

Outline

PI^2

CEM

CMA-ES

Outline

Reward-Weighted Averaging

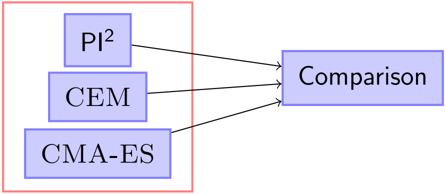
PI^2

CEM

CMA-ES

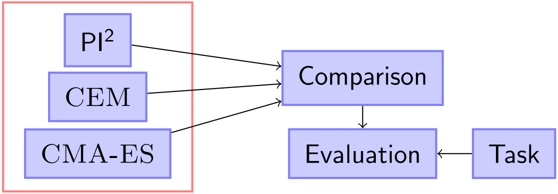
Outline

Reward-Weighted Averaging



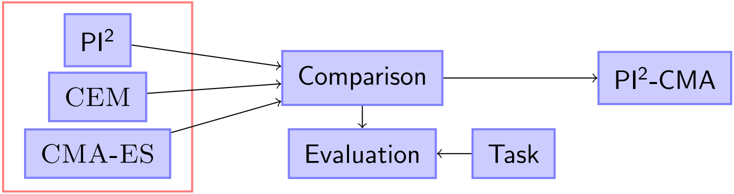
Outline

Reward-Weighted Averaging



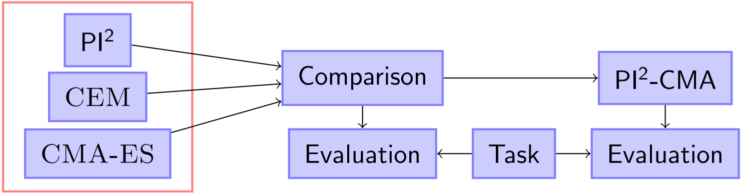
Outline

Reward-Weighted Averaging

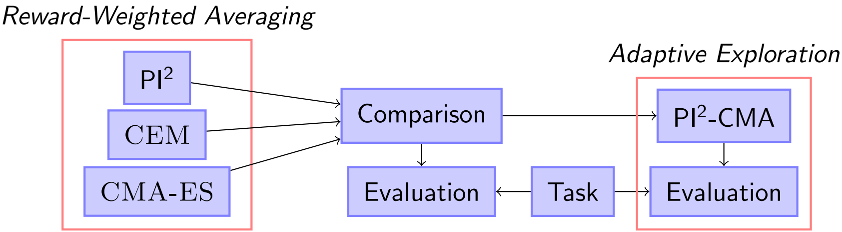


Outline

Reward-Weighted Averaging

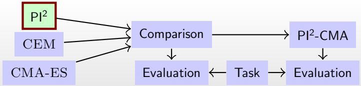


Outline



Reward-Weighted Averaging
PI^2
CEM
CMA-ES
Comparison
Evaluation
Task
Adaptive Exploration
PI^2-CMA
Evaluation

PI^2- Derivation Outline



- From first principles of Stochastic Optimal Control
    - Start with Hamilton Jacobi Bellman equations

1. Log transformation + benign assumption

   $\Rightarrow$ Linear!
2. Transform PDE to path integral with Feynman-Kac theorem

   $\Rightarrow$ Roll-outs!
3. Apply to parameterized policies

   $\Rightarrow$ Model free!

$\Rightarrow$ Iterative update rule for θ

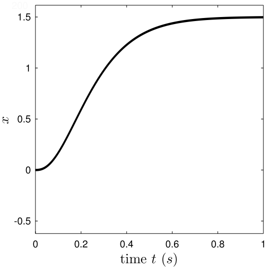
PI^2- Algorithm

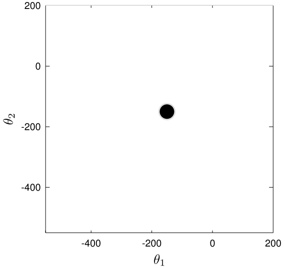
Words

Images

Formulae

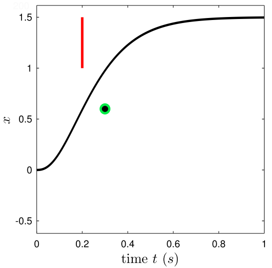
- **Input**: DMP with initial parameters $\boldsymbol{\theta}$

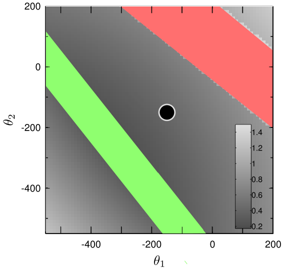




$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T \boldsymbol{\theta}$$

- **Input**: DMP with initial parameters $\boldsymbol{\theta}$, cost function J

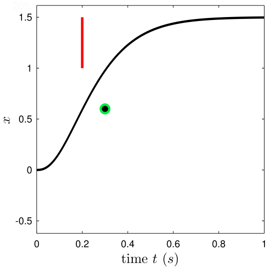


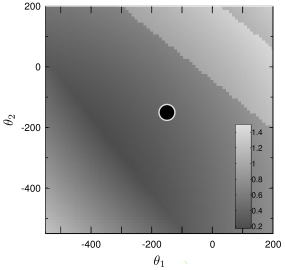


$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T \ \boldsymbol{\theta}$$

$$J(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} (q_t + \frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R} \boldsymbol{\theta}_t) \ dt$$

- **Input**: DMP with initial parameters $\boldsymbol{\theta}$, cost function J





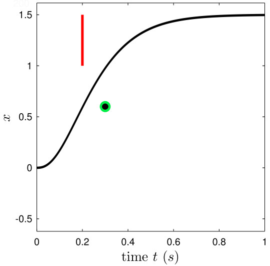
$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T\ \boldsymbol{\theta}$$

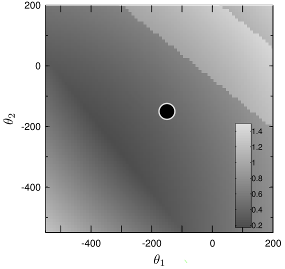
$$J(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} (q_t + \frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R}\boldsymbol{\theta}_t)\ dt$$

- **Input**: DMP with initial parameters $\boldsymbol{\theta}$, cost function J
- While (cost not converged)

Explore

Update





$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T \boldsymbol{\theta}$$

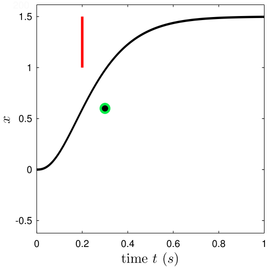
$$J(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} (q_t + \frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R}\boldsymbol{\theta}_t)\, dt$$

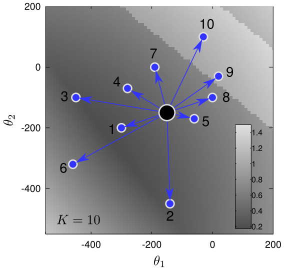
- **Input**: DMP with initial parameters $\boldsymbol{\theta}$, cost function J
- While (cost not converged)

Explore

sample exploration vectors

Update





$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{t,k})$$

$$J(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} (q_t + \frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R} \boldsymbol{\theta}_t)\, dt$$

$$\boldsymbol{\epsilon}_{t,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma}^{\epsilon})$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

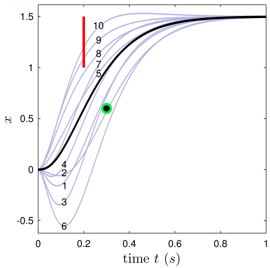
- **Input**: DMP with initial parameters $\boldsymbol{\theta}$, cost function J
- While (cost not converged)

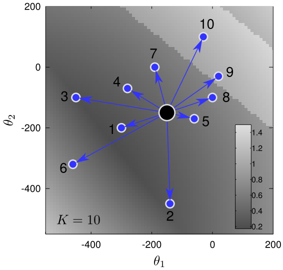
Explore

sample exploration vectors

execute DMP

Update





$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{t,k})$$

$$J(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N}(q_t + \frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R}\boldsymbol{\theta}_t)\, dt$$

$$\boldsymbol{\epsilon}_{t,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma}^{\epsilon})$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

- **Input**: DMP with initial parameters $\boldsymbol{\theta}$, cost function J
- While (cost not converged)

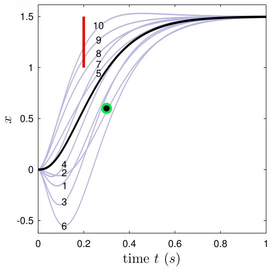
Explore

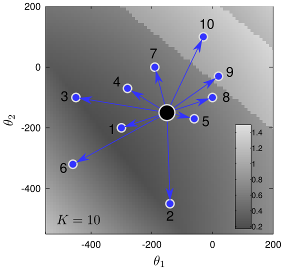
sample exploration vectors

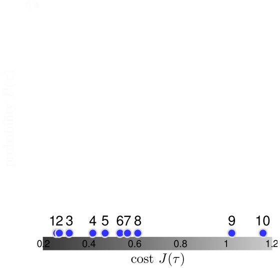
execute DMP

determine cost

Update







$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{t,k})$$

$$J(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} (q_t + \frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R}\boldsymbol{\theta}_t)\, dt$$

$$\boldsymbol{\epsilon}_{t,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma}^{\epsilon})$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

- **Input**: DMP with initial parameters $\boldsymbol{\theta}$, cost function J
- While (cost not converged)

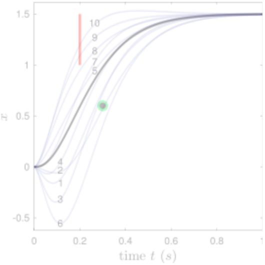
  Explore

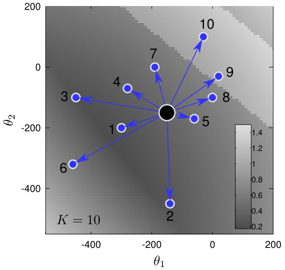
  sample exploration vectors

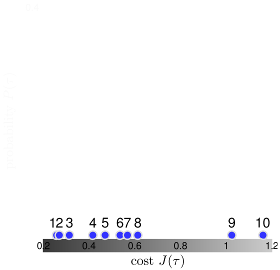
  execute DMP

  determine cost

Update







$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{t,k})$$

$$J(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} (q_t + \frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R} \boldsymbol{\theta}_t)\, dt$$

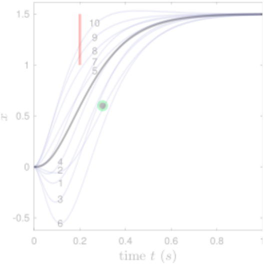
$$\boldsymbol{\epsilon}_{t,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma}^{\epsilon})$$

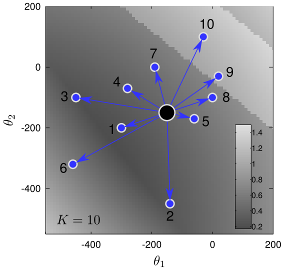
$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

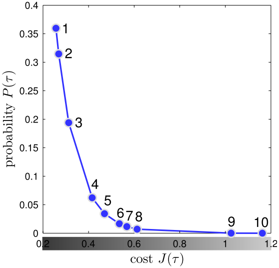
- **Input**: DMP with initial parameters $\boldsymbol{\theta}$, cost function J
- While (cost not converged)

Explore
sample exploration vectors
execute DMP
determine cost

Update
compute prob. from cost







$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{t,k})$$

$$J(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} (q_t + \frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R} \boldsymbol{\theta}_t)\, dt$$

$$\boldsymbol{\epsilon}_{t,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma}^{\epsilon})$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

$$P\left(\boldsymbol{\tau}_{i,k}\right) = \frac{e^{-\frac{1}{\lambda}J(\boldsymbol{\tau}_{i,k})}}{\sum_{k=1}^{K}[e^{-\frac{1}{\lambda}J(\boldsymbol{\tau}_{i,k})}]}$$

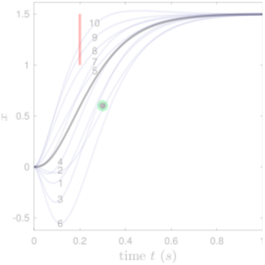
- **Input**: DMP with initial parameters $\boldsymbol{\theta}$, cost function J
- While (cost not converged)

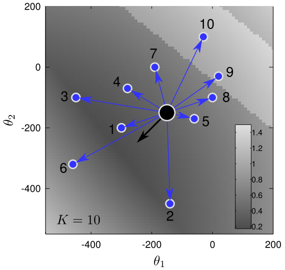
Explore

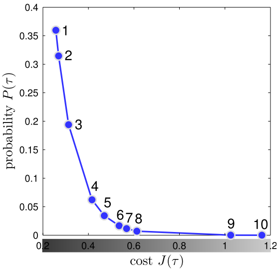
sample exploration vectors
execute DMP
determine cost

Update

compute prob. from cost
prob.-weighted averaging







$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{t,k})$$

$$J(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} (q_t + \frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R}\boldsymbol{\theta}_t)\, dt$$

$$\boldsymbol{\epsilon}_{t,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma}^{\epsilon})$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

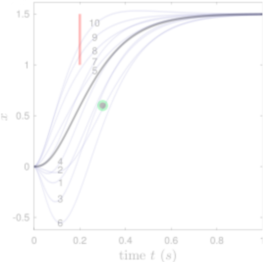
$$P\left(\boldsymbol{\tau}_{i,k}\right) = \frac{e^{-\frac{1}{\lambda}J(\boldsymbol{\tau}_{i,k})}}{\sum_{k=1}^{K}[e^{-\frac{1}{\lambda}J(\boldsymbol{\tau}_{i,k})}]}$$

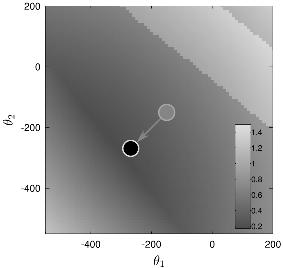
$$\delta\boldsymbol{\theta}_{t_i} = \sum_{k=1}^{K}\left[P\left(\boldsymbol{\tau}_{i,k}\right)\mathbf{M}_{t_i,k}\,\boldsymbol{\epsilon}_{t_i,k}\right]$$

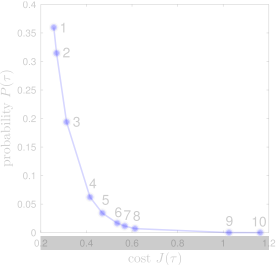
- **Input**: DMP with initial parameters $\boldsymbol{\theta}$, cost function J
- While (cost not converged)

Explore
sample exploration vectors
execute DMP
determine cost

Update
compute prob. from cost
prob.-weighted averaging
parameter update







$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{t,k})$$

$$J(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} (q_t + \frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R}\boldsymbol{\theta}_t)\, dt$$

$$\boldsymbol{\epsilon}_{t,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma}^{\epsilon})$$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta}$$

$$P\left(\boldsymbol{\tau}_{i,k}\right) = \frac{e^{-\frac{1}{\lambda}J(\boldsymbol{\tau}_{i,k})}}{\sum_{k=1}^{K}[e^{-\frac{1}{\lambda}J(\boldsymbol{\tau}_{i,k})}]}$$

$$\delta\boldsymbol{\theta}_{t_i} = \sum_{k=1}^{K}\left[P\left(\boldsymbol{\tau}_{i,k}\right)\mathbf{M}_{t_i,k}\,\boldsymbol{\epsilon}_{t_i,k}\right]$$

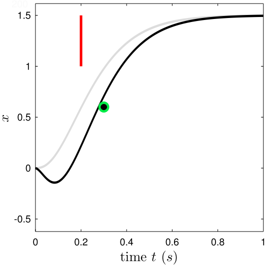
- **Input**: DMP with initial parameters $\boldsymbol{\theta}$, cost function J
- While (cost not converged)

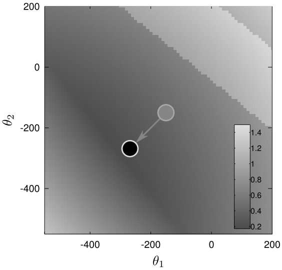
Explore

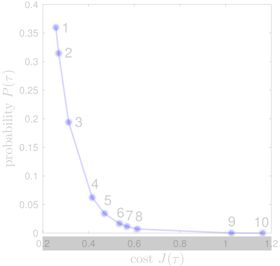
sample exploration vectors
execute DMP
determine cost

Update

compute prob. from cost
prob.-weighted averaging
parameter update







$$\frac{1}{\tau}\ddot{x}_t = \alpha(\beta(g - x_t) - \dot{x}_t) + \mathbf{g}_t^T(\boldsymbol{\theta} + \boldsymbol{\epsilon}_{t,k})$$

$$J(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} (q_t + \frac{1}{2}\boldsymbol{\theta}_t^T \mathbf{R} \boldsymbol{\theta}_t)\, dt$$

$$\boldsymbol{\epsilon}_{t,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma}^{\epsilon})$$

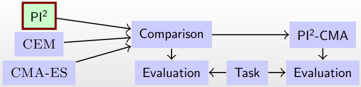
$$\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta}$$

$$P\left(\boldsymbol{\tau}_{i,k}\right) = \frac{e^{-\frac{1}{\lambda}J(\boldsymbol{\tau}_{i,k})}}{\sum_{k=1}^{K}[e^{-\frac{1}{\lambda}J(\boldsymbol{\tau}_{i,k})}]}$$

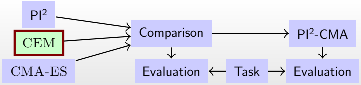
$$\delta\boldsymbol{\theta}_{t_i} = \sum_{k=1}^{K}\left[P\left(\boldsymbol{\tau}_{i,k}\right) \mathbf{M}_{t_i,k}\, \boldsymbol{\epsilon}_{t_i,k}\right]$$

PI^2- Algorithm

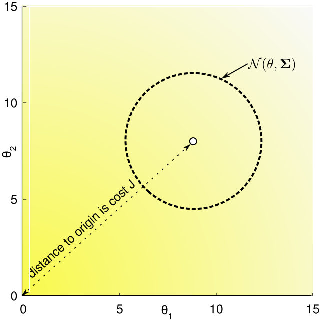


- Advantages
  - No gradient $\Rightarrow$ Deals with discontinous noisy cost functions
  - Update δ within convex hull of $\epsilon_{k=1\ldots K}$ $\Rightarrow$ Safe update rule
  - Arbitrary cost functions
  - Model-free
  - Fast convergence
  - Only one open parameter: magnitude of exploration ($\boldsymbol{\epsilon}_{t,k} \sim \mathcal{N}(0, \Sigma)$)
- Disadvantage
  - No global convergence guarantees...
  - Robotics: This is where imitation comes in! $\pi(\theta^{imit}) \approx \pi(\theta^*)$
- Applied to very high-dimensional, complex tasks...

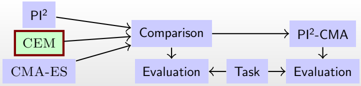
Cross-Entropy Method (CEM)



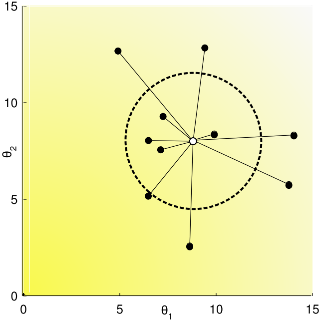
$\mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\Sigma})$



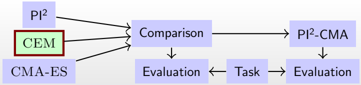
Cross-Entropy Method (CEM)



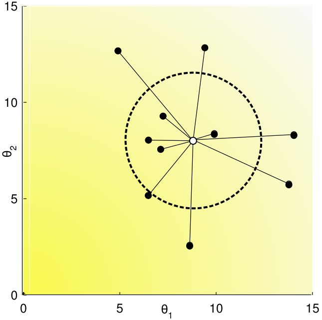
$\boldsymbol{\theta}_{k=1\ldots K} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$

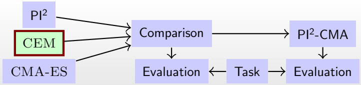


Cross-Entropy Method (CEM)



$$\boldsymbol{\theta}_{k=1\ldots K} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$$
$$\forall k \; J_k = J(\boldsymbol{\theta}_k)$$



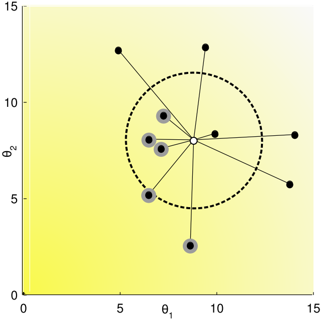


Cross-Entropy Method (CEM)

$$\boldsymbol{\theta}_{k=1\dots K} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$$

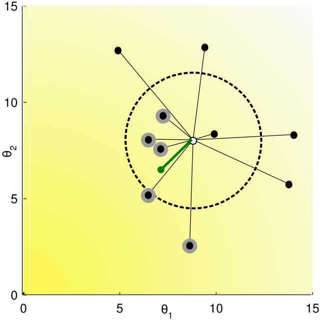
$$\forall k \; J_k = J(\boldsymbol{\theta}_k)$$

$$\boldsymbol{\theta}_{k=1\dots K} \leftarrow \texttt{sort } \boldsymbol{\theta}_{k=1\dots K} \texttt{ w.r.t } J_{k=1\dots K}$$



Cross-Entropy Method (CEM)

$$\boldsymbol{\theta}_{k=1\dots K} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$$
$$\forall k \; J_k = J(\boldsymbol{\theta}_k)$$
$$\boldsymbol{\theta}_{k=1\dots K} \leftarrow \texttt{sort}\; \boldsymbol{\theta}_{k=1\dots K} \;\texttt{w.r.t}\; J_{k=1\dots K}$$
$$\boldsymbol{\theta}^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} \boldsymbol{\theta}_k$$



Cross-Entropy Method (CEM)

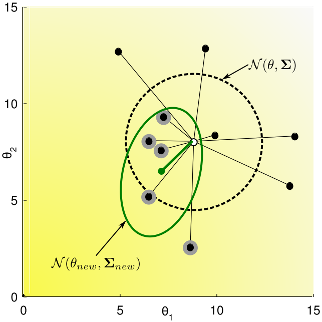
$$\boldsymbol{\theta}_{k=1\ldots K} \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\Sigma})$$

$$\forall k \; J_k = J(\boldsymbol{\theta}_k)$$

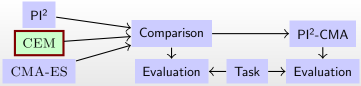
$$\boldsymbol{\theta}_{k=1\ldots K} \leftarrow \texttt{sort } \boldsymbol{\theta}_{k=1\ldots K} \texttt{ w.r.t } J_{k=1\ldots K}$$

$$\boldsymbol{\theta}^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} \boldsymbol{\theta}_k$$

$$\boldsymbol{\Sigma}^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} (\boldsymbol{\theta}_k - \boldsymbol{\theta})(\boldsymbol{\theta}_k - \boldsymbol{\theta})^{\intercal}$$



Cross-Entropy Method (CEM)



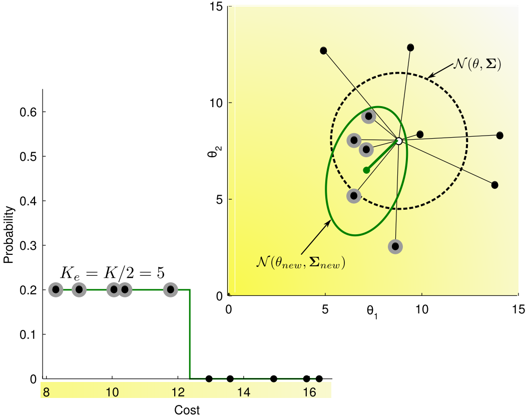
$$\boldsymbol{\theta}_{k=1\dots K} \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\Sigma})$$

$$\forall k \;\; J_k = J(\boldsymbol{\theta}_k)$$

$$\boldsymbol{\theta}_{k=1\dots K} \leftarrow \texttt{sort}\ \boldsymbol{\theta}_{k=1\dots K}\ \texttt{w.r.t}\ J_{k=1\dots K}$$

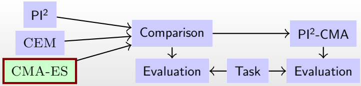
$$\boldsymbol{\theta}^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} \boldsymbol{\theta}_k$$

$$\boldsymbol{\Sigma}^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} (\boldsymbol{\theta}_k - \boldsymbol{\theta})(\boldsymbol{\theta}_k - \boldsymbol{\theta})^\intercal$$



- This algorithm can be interpreted as performing reward-weighted averaging

CMA-ES



- Covariance Matrix Adaptation - Evolutionary Strategy
- Like CEM, but
  - Different mapping from cost to probability
  - More sophisticated method for updating covariance matrix:

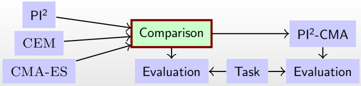
$$p_\sigma \leftarrow (1 - c_\sigma)\, p_\sigma + \sqrt{c_\sigma (2 - c_\sigma) \mu_P \Sigma^{-1}}\, \frac{\boldsymbol{\theta}^{new} - \boldsymbol{\theta}}{\sigma} \tag{1}$$

$$\sigma_{new} = \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma\|}{E\|\mathcal{N}(0, I)\|} - 1\right)\right) \tag{2}$$

$$p_\Sigma \leftarrow (1 - c_\Sigma)\, p_\Sigma + h_\sigma \sqrt{c_\Sigma (2 - c_\Sigma) \mu_P}\, \frac{\boldsymbol{\theta}^{new} - \boldsymbol{\theta}}{\sigma} \tag{3}$$

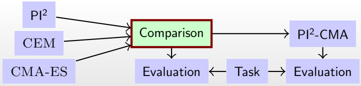
$$\Sigma^{new} = (1 - c_1 - c_\mu)\, \Sigma + c_1 (p_\Sigma p_\Sigma^T + \delta(h_\sigma)\Sigma) + c_\mu \sum_{k=1}^{K_e} P_k (\boldsymbol{\theta}_k - \boldsymbol{\theta})(\boldsymbol{\theta}_k - \boldsymbol{\theta})^\intercal \tag{4}$$

PI^2/CEM/CMA-ES- Similarities



- PI^2/CEM/CMA-ES are all based on
  - Exploration: sample from a Gaussian $\boldsymbol{\theta}_k \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$
  - Parameter update: Reward-Weighted Averaging $\boldsymbol{\theta}^{new} = \sum_{k=1}^{K} P_k \boldsymbol{\theta}_k$

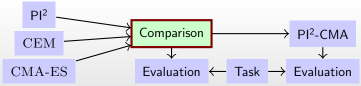
PI^2/CEM/CMA-ES- Similarities



- PI^2/CEM/CMA-ES are all based on
  - Exploration: sample from a Gaussian $\boldsymbol{\theta}_k \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$
  - Parameter update: Reward-Weighted Averaging $\boldsymbol{\theta}^{new} = \sum_{k=1}^{K} P_k \boldsymbol{\theta}_k$

Similarity striking, as algorithms derived from very different principles!

PI^2/CEM/CMA-ES- Similarities



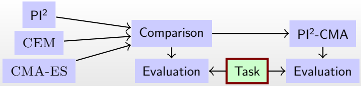
- PI^2/CEM/CMA-ES are all based on
  - Exploration: sample from a Gaussian $\quad \boldsymbol{\theta}_k \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$
  - Parameter update: Reward-Weighted Averaging $\quad \boldsymbol{\theta}^{new} = \sum_{k=1}^{K} P_k \boldsymbol{\theta}_k$

Similarity striking, as algorithms derived from very different principles!

- CEM is a special case of CMA-ES: proof in paper.

  (maybe this was already known?)

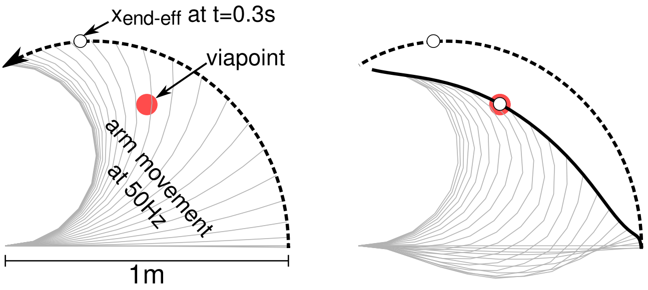
PI^2/CEM/CMA-ES- Differences



- Also some differences
- Evaluated on following task:

PI^2/CEM/CMA-ES- Differences

- Also some differences
- Evaluated on following task:



$$J(\boldsymbol{\tau}_t) = \delta(t-0.3)\cdot((x_t-0.5)^2+(y_t-0.5)^2) + \frac{\sum_{d=1}^{D}(D+1-d)(\ddot{a}_t)^2}{\sum_{d=1}^{D}(D+1-d)} \quad (5)$$

PI^2/CEM/CMA-ES- Differences

Exploration Noise

PI^2

$$\boldsymbol{\theta}_{k,t} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$$

CEM

$$\boldsymbol{\theta}_{k} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$$

CMA-ES

$$\boldsymbol{\theta}_{k} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$$

PI^2/CEM/CMA-ES- Differences

Exploration Noise

PI^2

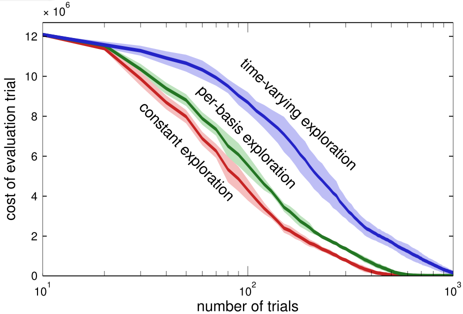
$$\boldsymbol{\theta}_{k,t} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$$

CEM

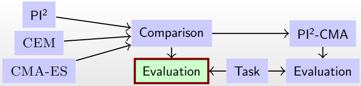
$$\boldsymbol{\theta}_{k} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$$

CMA-ES

$$\boldsymbol{\theta}_{k} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$$

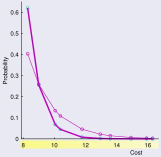


PI^2/CEM/CMA-ES- Differences

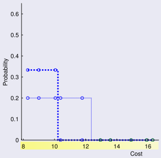


Eliteness

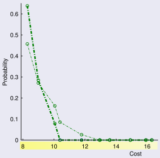
PI^2



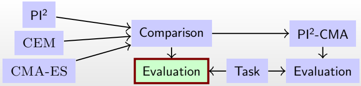
CEM



CMA-ES

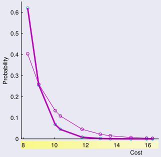


PI^2/CEM/CMA-ES- Differences

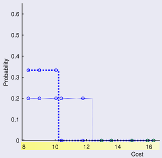


Eliteness

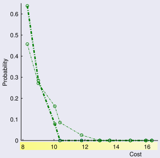
PI^2

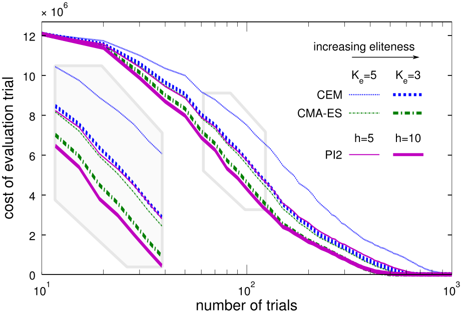


CEM



CMA-ES



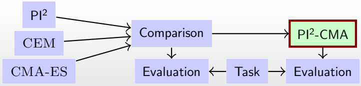


PI^2/CEM/CMA-ES- Differences

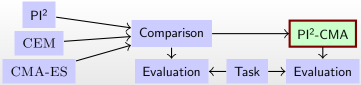
Covariance Matrix Updating

PI^2	CEM	CMA-ES
No	Yes (simple)	Yes (sophisticated)

PI^2-CMA

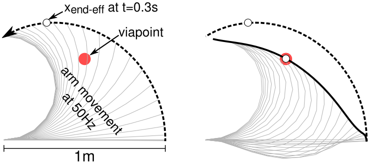


- Suggests a new algorithm: PI^2-CMA
  - Constant exploration noise
  - Eliteness measure from PI^2
  - Covariance matrix updating from CEM/CMA-ES

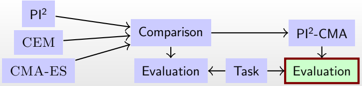


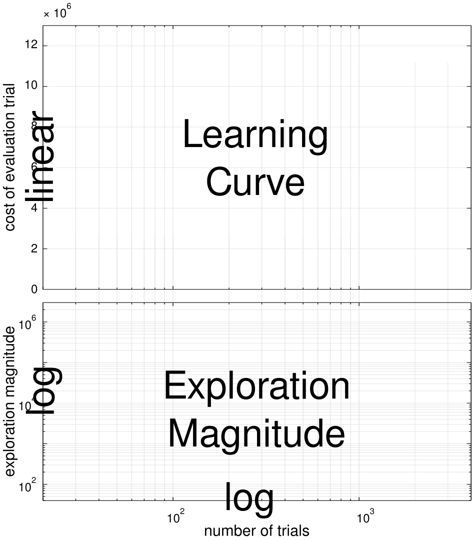
PI^2-CMA

- Suggests a new algorithm: PI^2-CMA
  - Constant exploration noise
  - Eliteness measure from PI^2
  - Covariance matrix updating from CEM/CMA-ES
- Basically PI^2, but with adaptive exploration
  - In PI^2, exploration magnitude must be tuned by hand
  - Next evaluation demonstrates advantages of PI^2-CMA

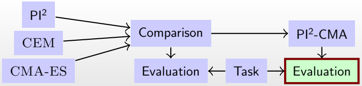


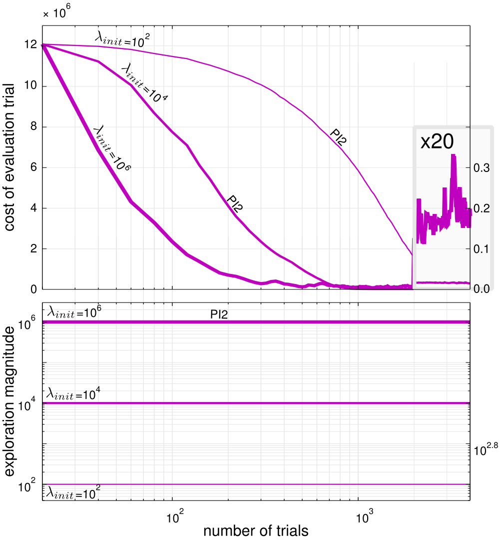
PI^2-CMA- Adaptive Exploration





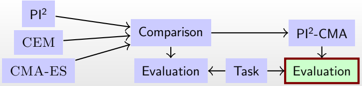
PI²-CMA- Adaptive Exploration

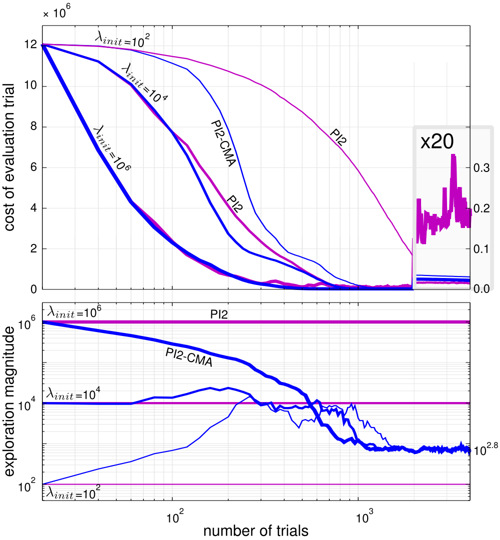




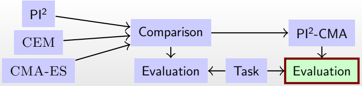
⇒ Exploration magnitude influences convergence speed and exploitation

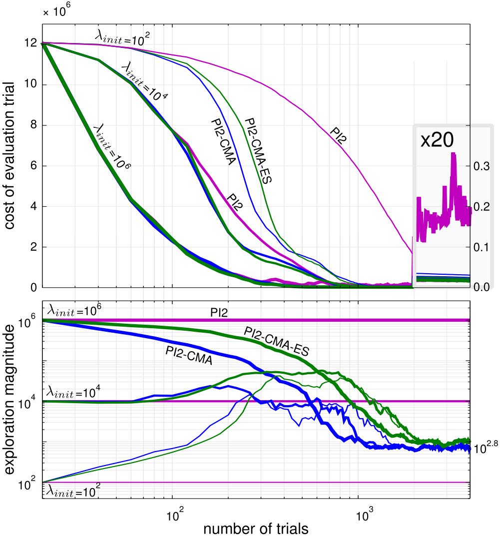
PI^2-CMA- Adaptive Exploration





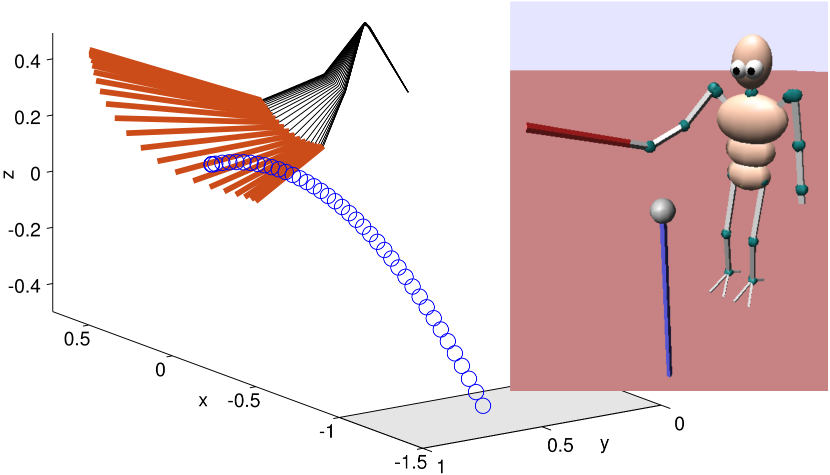
PI^2-CMA- Adaptive Exploration



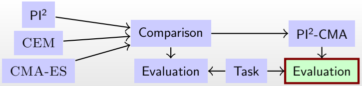


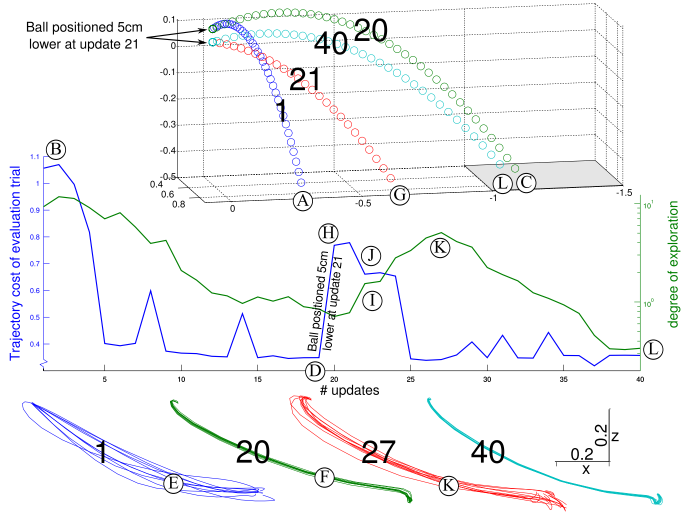
PI²-CMA- Adaptive Exploration

More recent results (not in paper)

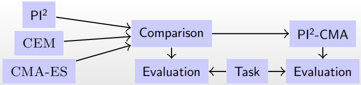


PI^2-CMA- Adaptive Exploration

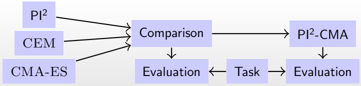




Summary

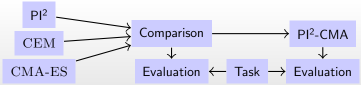


- PI^2/CEM/CMA-ES have identical update rules: reward-weighted averaging
- Apply covariance matrix adaptation (as in CEM/CMA-ES) to PI^2
- Novel algorithm PI^2-CMA
    - With adaptive exploration (other algorithmic parameters trivial to tune)
- Future work
    - Further analysis and theoretical validation
    - Evaluation on real robots



Thank you for your attention!
Questions?

Bibliography



Stulp, F. and Schaal, S. (2011).
Hierarchical reinforcement learning with motion primitives.
In *11th IEEE-RAS International Conference on Humanoid Robots*.

Stulp, F., Theodorou, E., Buchli, J., and Schaal, S. (2011a).
Learning to grasp under uncertainty.
In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Stulp, F., Theodorou, E., Kalakrishnan, M., Pastor, P., Righetti, L., and Schaal, S. (2011b).
Learning motion primitive goals for robust manipulation.
In *International Conference on Intelligent Robots and Systems (IROS)*.