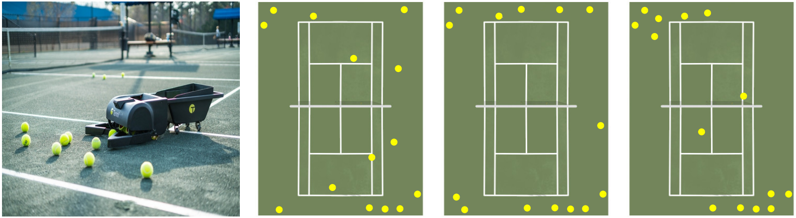
# From Policy Gradient to Actor-Critic methods

## The Policy Search problem

Olivier Sigaud

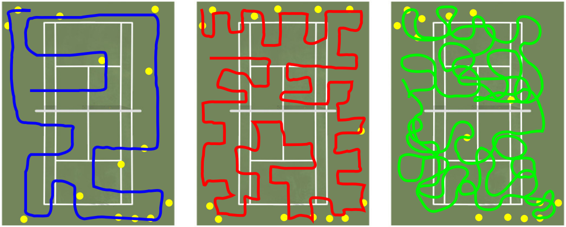
Sorbonne Université
http://people.isir.upmc.fr/sigaud

## Example: a (cheap) tennis ball collector



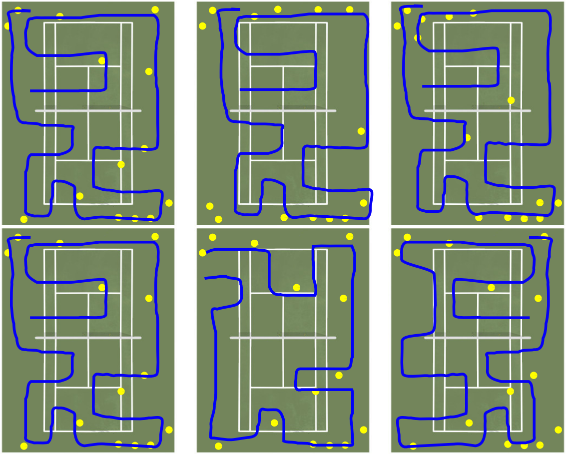
- A robot without a ball sensor
- Travels on a tennis court based on a parametrized controller
- Performance: number of balls collected in a given time
- Just depends on robot trajectories and ball positions

## Influence of policy parameters



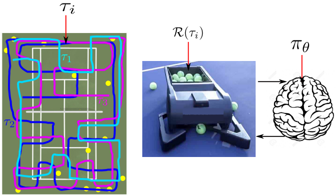
- Controller parameters: proba of turn per time step, travelling speed
- How do the parameters influence the performance?
- The performance depends on the spread of balls → need to repeat
- Policy search: find the optimal policy parameters

## Two sources of stochasticity



- From the environment: position of the balls
- From the policy, if it is stochastic

## The policy search problem: formalization

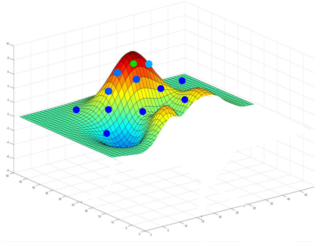



- Let:
  - $\pi_{\boldsymbol{\theta}}$ be the parametrized policy of the robot
  - $\tau_i$ is a robot trajectory
  - $R(\tau_i)$ is the corresponding return
  - $J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}}[R(\tau)]$ is the global utility (or cost) function
- We have to sample the expectation, thus the goal is to find

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, J(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{\tau} P(\tau, \boldsymbol{\theta}) R(\tau) \quad (1)$$

Deisenroth, M. P., Neumann, G., Peters, J., et al. (2013) A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142

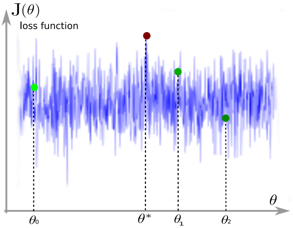
## Direct Policy Search is black box optimization



- $J(\boldsymbol{\theta})$ is the performance over policy parameters
- Choose a $\boldsymbol{\theta}$
- Generate trajectories $\tau_{\boldsymbol{\theta}}$
- Get the return $J(\boldsymbol{\theta})$ of these trajectories
- Look for a better $\boldsymbol{\theta}$, repeat

- DPS uses $(\boldsymbol{\theta}, J(\boldsymbol{\theta}))$ pairs and directly looks for $\boldsymbol{\theta}$ with the highest $J(\boldsymbol{\theta})$

## (Truly) Random Search



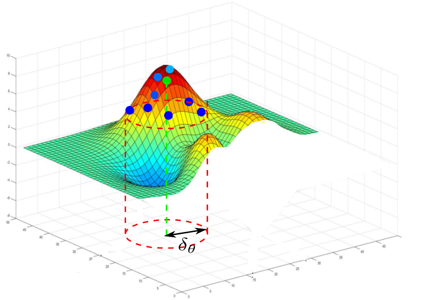

- Select $\boldsymbol{\theta}_i$ randomly
- Evaluate $J(\boldsymbol{\theta}_i)$
- If $J(\boldsymbol{\theta}_i)$ is the best so far, keep $\boldsymbol{\theta}_i$
- Loop until $J(\boldsymbol{\theta}_i) > target$ (maximize reward)

- Of course, this is not efficient if the space of $\boldsymbol{\theta}$ is large
- General "blind" algorithm, no assumption on $J(\boldsymbol{\theta})$
- We can do better if $J(\boldsymbol{\theta})$ shows some local regularity

Sigaud, O. & Stulp, F. (2019) Policy search in continuous action domains: an overview. *Neural Networks*, 113:28-40

## Direct policy search

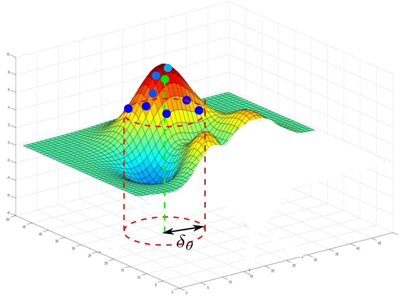
- Locality assumption: The function is locally smooth, unknown good solutions are close to known good solutions



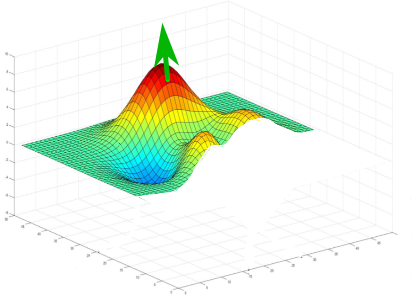

Variation - selection

- Variation - selection: Perform well chosen variations, evaluate them
- Locality generally controlled using a multivariate Gaussian

## Gradient ascent



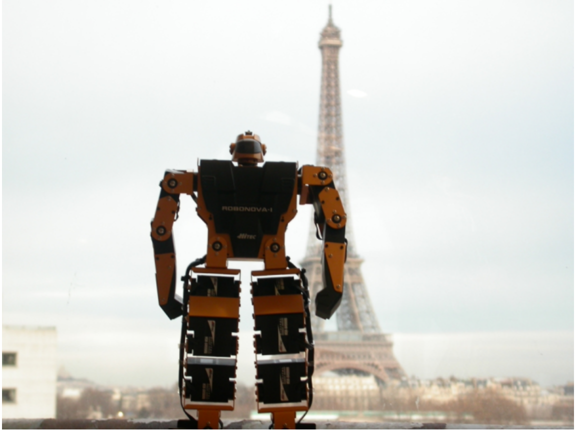

Variation - selection



Gradient ascent

- Gradient ascent: Following the gradient from analytical knowledge
- Issue: in general, the function $J(\boldsymbol{\theta})$ is unknown
- How can we apply gradient ascent without knowing the function?
- The answer is the Policy Gradient Theorem
- Next lesson: Policy Gradient

## Any question?




Send mail to: Olivier.Sigaud@upmc.fr

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al.
A survey on policy search for robotics.
*Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

Olivier Sigaud and Freek Stulp.
Policy search in continuous action domains: an overview.
*Neural Networks*, 113:28–40, 2019.