Learning to grasp with a jamming gripper

Debaleena MISRA

Advisors: Dr. Jean-Baptiste Mouret, Dr. Olivier Kermorgant

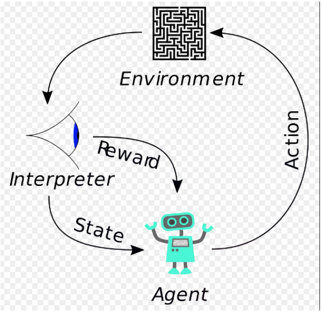
27 Aug 2018

Outline

1. Introduction
2. Objectives of thesis
3. State-of-art
    - Basics of Reinforcement Learning
    - Reinforcement Learning in Robotics
    - Black-Box Data-Efficient Policy Search for Robotics (Black-DROPS)
4. Experimental Setup
5. Contribution
6. Results
7. Conclusion

INTRODUCTION

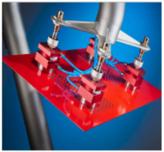
Motivation



Why use reinforcement learning in robotics?

- No need for complete model information of the robot
- Adaptation to dynamic environment changes
- Recovery from unknown damages by finding compensatory behaviors

Grasping in robotics

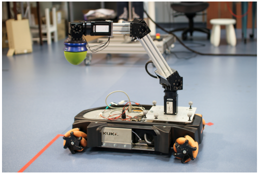


Objectives

1. Implement a data-efficient reinforcement learning algorithm called *Black-box Data-Efficient Policy Search for Robotics*[1] for grasping by a jamming gripper.
2. Vision-based object detection using depth cameras

Challenges:

- Can Black-DROPS be scaled to complex realistic robotics tasks?
- What is a suitable design of reward function?
- Complexities of the jamming gripper



[1] Konstantinos Chatzilygeroudis et al. "Black-box data-efficient policy search for robotics". In: *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. 2017, pp. 51–58.

STATE-OF-ART

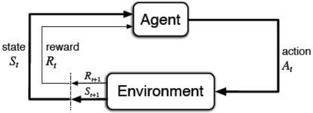
Reinforcement Learning

Agent interacts with environment to decide optimal behaviour.

- Feedback in the form of **rewards**
- **Goal** - Learn to take actions that maximizes total future reward

Markov property Action outcome only depends on current state

- States **S**
- Actions **A**
- Transition Probabilities $\mathbf{T}(\mathbf{s_t}, \mathbf{a_t}, \mathbf{s_{t+1}})$
- Reward Function $\mathbf{R}(\mathbf{s_t}, \mathbf{a_t})$



Needs to find optimal **policy** to map states to actions and maximize expected long-term cumulative reward

RL is a Markov Decision Process with **T** and/or **R** unknown

- Dont know which states are good and what actions to take
- Must explore states and actions to learn this knowledge

Robot reinforcement learning

Challenges

- High cost of robot interactions with environment
- Data-efficiency and real-time requirements
- High-dimensional continuous state-action space

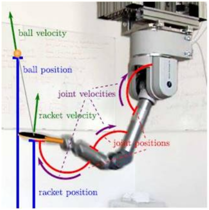


Fig: Ball-paddling RL[2]

Approaches

− **Value-function based RL** explores entire state-space, thus struggle with high-dimensionality and cannot be used

+ **Policy search** preferred in robot RL due to ease of scalability to high-dimensions and incorporation of expert knowledge (by both structure or initialisation)

2

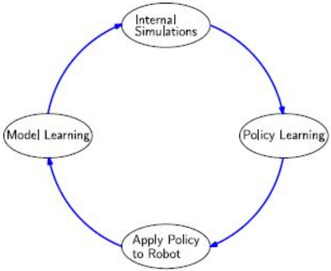
[2] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. "A survey on policy search for robotics". In: *Foundations and Trends® in Robotics* 2.1–2 (2013), pp. 1–142

Model-based Policy Searches

+ More data-efficient than model-free ones
+ More complex policies can be optimized
− Modelling errors can lead to very poor outcomes

Steps

1. Learn dynamic models from data-set
   - Gaussian Processes
   - Bayesian Locally Weighted Regression
   - Time-dependent linear models
2. Use Learned Model as Simulator for policy optimization
3. Update Policy



Black-DROPS[3] Algorithm

+ Model-based policy search

+ Highly data-efficient

+ No constraint on reward function

+ No restriction on policy representation (any parametrized policy maybe used)

+ Accounts for uncertainty in dynamics modelling

+ Performs more global search than gradient-based methods

[3] Konstantinos Chatzilygeroudis et al. "Black-box data-efficient policy search for robotics". In: *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. 2017, pp. 51–58.

Black-DROPS Algorithm: Problem formulation

Dynamical systems of form: $\mathbf{x}_{t+1} = \mathbf{x}_t + f(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{w}$

where **x, u, f, w** are the states, controls, Gaussian system noise and unknown transition dynamics respectively.

Objective: To find a deterministic policy π, $\mathbf{u} = \pi(\mathbf{x}|\boldsymbol{\theta})$, that maximizes the *expected long-term reward*:

$$J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=1}^{T} r(\mathbf{x}_t)\middle|\boldsymbol{\theta}\right] \quad (1)$$

$r(\mathbf{x}_t)$: Immediate reward of being in state $\mathbf{x}$ at time t.

(1) Dynamics model learning using Gaussian Process

Model $\hat{f}$ approximates unknown system dynamics f

Input tuples: $\tilde{x}_t = (x_t, u_t) \in \mathbb{R}^{E+U}$ Training targets: $\triangle_{x_t} = x_{t+1} - x_t \in \mathbb{R}^E$

E independent GPs used to model each dimension of the difference vector

GP computed (with $k_{\hat{f}_d}$ as the kernel function):

$$\hat{f}_d(\tilde{x}) \sim \mathcal{GP}\left(\mu_{\hat{f}_d}(\tilde{x}), k_{\hat{f}_d}(\tilde{x}, \tilde{x}')\right)$$

Let $D^d_{1:t} = f_d(\tilde{x_1}), ..., f_d(\tilde{x_t})$ be a set of observations

Then the GP can be queried at a new input point $\tilde{x}_*$ as:

$$p(\hat{f}_d(\tilde{x}_*)|D_{1:t}, \tilde{x}_*) = \mathcal{N}(\mu_{\hat{f}_d}(\tilde{x}_*), \sigma^2_{\hat{f}_d}(\tilde{x}_*))$$

Mean $\mu_{\hat{f}_d}(\tilde{x}_*) = \mathbf{k}^T_{\hat{f}_d} K^{-1}_{\hat{f}_d} D_{1:t}$ Variance $\sigma^2_{\hat{f}_d} = k_{\hat{f}_d}(\tilde{x}_*, \tilde{x}_*) - \mathbf{k}^T_{\hat{f}_d} K^{-1}_{\hat{f}_d} \mathbf{k}_{\hat{f}_d}$

Black-DROPS Algorithm: Approach

Kernel vector: $\mathbf{k} = k(D_{1:t}, \tilde{x}_*)$, Kernel matrix: K with elements $K^{ij} = k(\tilde{x}_i, \tilde{x}_j)$
Exponential kernel with automatic relevance determination[4] is used:

$$k_{\hat{f}_d}(\tilde{x_p}, \tilde{x_q}) = \sigma_d^2 \exp\left(-\frac{1}{2}(\tilde{x_p} - \tilde{x_q})^T\right) \Lambda_d^{-1}(\tilde{x_p} - \tilde{x_q}) + \delta_{pq}\sigma_{n_d}^2$$

$[\Lambda_d, \sigma_d^2, \sigma_{nd}^2]$ = Kernel hyperparameters, $\delta_{pq} = 1$ when $p = q$ and 0 otherwise

(2) Learn immediate reward function with a GP

A reward $r(x) \in \mathbb{R}$ for each state x:

$$\hat{r}(\tilde{x}) \sim \mathcal{GP}(\mu_r(\tilde{x}), k_r(\tilde{x}, \tilde{x}'))$$

[4] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian process for machine learning*. MIT press, 2006.

(3) Policy Evaluation

Let $G(\boldsymbol{\theta})$ be the measurement of $J(\boldsymbol{\theta})$ perturbed by noise $N(\boldsymbol{\theta})$. Then

$$\mathbb{E}\Big[G(\boldsymbol{\theta})\Big] = J(\boldsymbol{\theta}) + \mathbb{E}\Big[N(\boldsymbol{\theta})\Big]$$

Assuming $\mathbb{E}[N(\boldsymbol{\theta})] = 0$ for all $\boldsymbol{\theta} \in \mathbb{R}^{\theta}$, maximizing $\mathbb{E}[G(\boldsymbol{\theta})]$ becomes equivalent to maximizing $J(\boldsymbol{\theta})$.

(4) Policy Search

Black-box optimizer for noisy functions - **CMA-ES**[5]

[5] Nikolaus Hansen. "Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed". In: *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. ACM. 2009, pp. 2389–2396.

Black-DROPS Algorithm: Summary

1. Record state-action sequence from initial random policy.
2. Learn a probabilistic model of the dynamics using Gaussian Process (GP) regression.
3. Use the model uncertainties in black-box optimization (using CMA-ES) of the policy.
4. Execute the policy on the robot and collect new state-action data.
5. Repeat from (2) until the task is solved.

Black-DROPS with Priors[6]

- Policy initialization done on prior model in simulations
- Real data used to explicitly learn unknown dynamics
- Total number of roll-outs greatly reduced

Gaussian Process with Simulator as Mean function

Dynamical systems of form: $x_{t+1} = x_t + M(x_t, u_t, \phi_M) + f(x_t, u_t, \phi_K) + w$

Input tuples: $\tilde{x}_t = (x_t, u_t) \in \mathbb{R}^{E+U}$ Training targets: $\triangle_{x_t} = x_{t+1} - x_t \in \mathbb{R}^E$

E independent GPs used to model each dimension of the difference vector

GP can be queried at a new input point $\tilde{x}_*$ as:

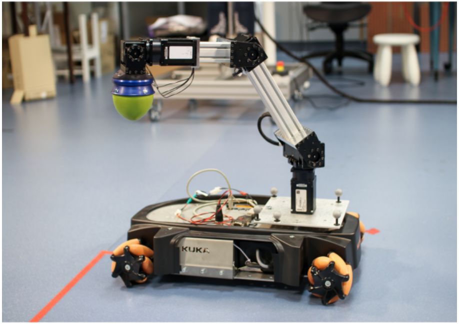
$$p(\hat{F}(\tilde{x}_*)|D_{i:t}, \tilde{x}_*) = \mathcal{N}(\mu(\tilde{x}_*), \sigma^2(\tilde{x}_*))$$

$$\mu(\tilde{x}_*) = M(\tilde{x}_*) + \mathbf{k}^T K^{-1}(D_{1:t} - M(\tilde{x}_*)) \qquad \sigma^2 = k(\tilde{x}_*, \tilde{x}_*) - \mathbf{k}^T K^{-1}\mathbf{k}$$

where $M(\tilde{x}_*)$ is the simulator function with initial guess of system dynamics

[6] Konstantinos Chatzilygeroudis and Jean-Baptiste Mouret. "Using Parameterized Black-Box Priors to Scale Up Model-Based Policy Search for Robotics". In: *International Conference on Robotics and Automation (ICRA)* (2018).

Experimental set-up



CONTRIBUTION

Work-flow

1. Identify object location
2. Use Black-DROPS to learn to reach the object in correct orientation from vertically above
3. Apply time-based pneumatic control to the versaball for the grasping action

Implementation of Black-DROPS

Design of a parametrized policy to describe the system and grasping task

- **States**: Joint positions, joint velocities and time
  $x_{arm} = [q_0, q_1, q_2, q_3, q_4, v_0, v_1, v_2, v_3, v_4, t]\epsilon\mathbb{R}^{11}$
  where $x_0 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$
- **Actions**: Joint velocities
  $u_{arm} = [v_0, v_1, v_2, v_3, v_4]\epsilon\mathbb{R}^5$
  where $-1.0 \leq v_i \leq 1.0\ rad/s,\ i = 0, 1, 2, 3, 4$
- **Policy**: A feedforward neural network.
  - i^{th} layer has network function $y_i = \phi_i(W_i y_{i-1} + b_i)$
    (where W_i=weight matrix, b_i=bias vector, ϕ_i=activation function
    y_{i-1}=input, y_i=output vector)
  - 1 hidden layer
  - Hyperbolic tangent activation function
  - Representation : $\pi(x) = u_{max} y_1 = u_{max} \phi\left(W_1 y_0 + b_1\right)$
    and $y_0 = \phi\left(W_0 x + b_0\right)$ (where x is the input state vector)

Reward Computation

- Saturated and distance-based reward function used.
- Novelty in identifying factors contributing to successful grasp
  - **Distance to object location** : $r_{goal}(x) = exp\left(-\frac{1}{2\sigma_c^2}\|p_x - p_*\|\right)$
    (p_x=gripper position in state x, p_*=target object location)
  - **Deviation from desired angle** : $r_{angle}(x) = exp\left(-\frac{1}{2\sigma_c^2}\|c_x - c_*\|\right)$
    (c_x=gripper orientation in state x, c_*=target cosine for desired angle between gripper and vertical axis)
  - **Penalising effect of drastic actions** $r_{actions}(x) = \frac{\|u_{arm}\|}{\|u_{arm}\|_{max}}$
    (u_{arm}=current actions, $\|u_{arm}\|_{max}$=normalising factor)

Total reward $r_{arm}(x) \in [0,1]$ is formulated as a weighted sum of 3 factors:

$$r_{arm}(x) = w_1 * r_{goal}(x) + w_2 * r_{angle}(x) - w_3 * r_{actions}(x)$$

Reward Computation

- Gripper must approach object from **vertically above** to align centrally along object's axis
- Apply domain knowledge in form of sub-goals in learning scheme by **reward shaping**

During an episode of duration T, at any current time t:

if $(t < T/2)$ **then**
$r_{goal}(x) = w_1 * exp\left(-\frac{1}{2\sigma_c^2}||p_x - p_{subgoal}||\right)$
else
$r_{goal}(x) = w_1 * exp\left(-\frac{1}{2\sigma_c^2}||p_x - p_{actualgoal}||\right)$
end if

Control of the versaball

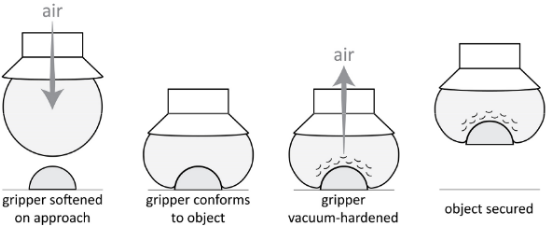


Fig: Grasping by an universal jamming gripper

Pneumatic control to implement pressure transitions by:

- 2 pumps and 2 electro-valves, one each for air inflow and suction
- Devices are successively switched on and off using an USB-controlled relay board

Visual object detection

Perception algorithm using PCL[7] library functions :

1. Point cloud information of the scene
2. Voxelgrid filtering
3. Planar segmentation by RANSAC
4. Removal of plane inliers
5. Euclidean clustering
6. Cluster centroid computation

[7] http : //www.pointclouds.org/,

RESULTS

Results: Object detection

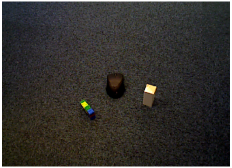


Fig: (1)RGB view of scene

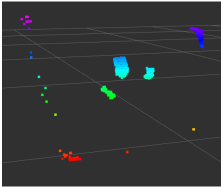


Fig: (3)Plane segmented

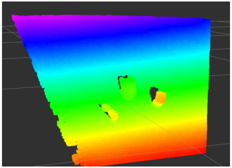


Fig: (2)Depth view of scene

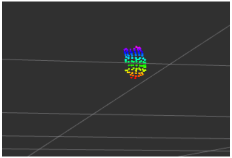


Fig: (4)Cluster of an object

Results: Black-DROPS (in simulation)

- **Task** : Reach object with gripper at 90° to it, and approach from top
- **State-action space :** 11D
- Working policy found within **6-7 episodes**
- Episode duration : **4 sec**
- Average computation time per episode : **2-3 min**
- Median behavior of 20 replicates shown

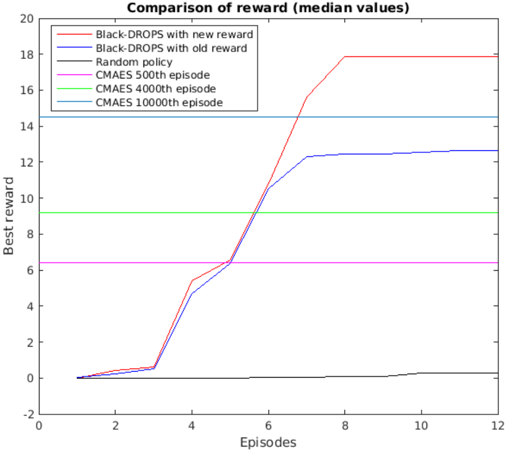


Fig: Reward comparison

Results: Black-DROPS (in simulation)

Significance of angular component in the reward formulation seen below in comparison with former and current reward settings of Black-DROPS

Number of replicates : **20**

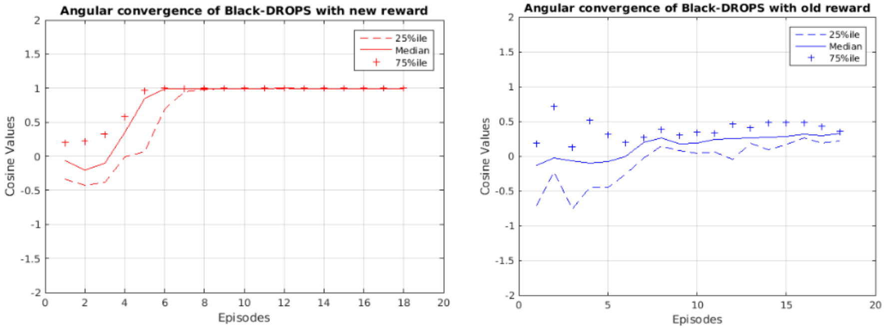


Fig: Angular convergence comparison

Results: Black-DROPS (on real robot)

- **Task** : Reach object with gripper at 90° to it, and approach from top
- **State-action space :** 11D
- Working policy found within **6-7 episodes**
- Episode duration : **4 sec**
- Average computation time per episode : **2-3 min**
- Simulation behavior validated on real system

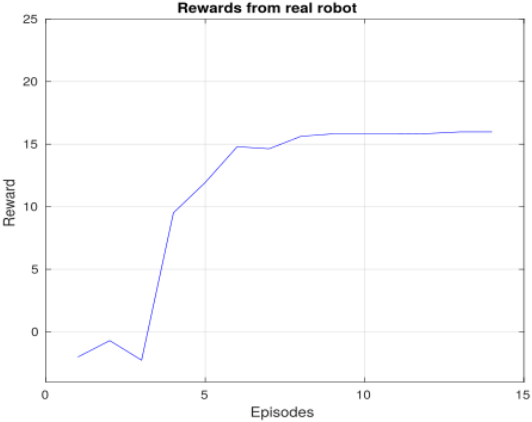


Fig: Rewards from experiment on real robot

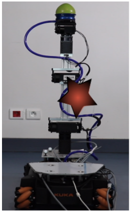
Damaged conditions

Performance so far

- Black-DROPS adapted successfully for the proposed task on the real system
- Model-learning is fast and efficient, and working policy is found in few trials

Why learning is essential

- What happens if robot suffers unknown damages?
- Is quick recovery possible without diagnosing the damages?



Next section

Testing of Black-DROPS and its variant (using priors on the model for policy initialisation) on a damaged setting of the robot

Results: Black-DROPS with priors

- **Task** : Reach object with gripper at 90° to it, and approach from top
- Artificial damages induced by sending **inaccurate (reduced) velocity commands** to the actuators
- Working policy found by both variants but in fewer trials **(3-4)** when priors on model is used

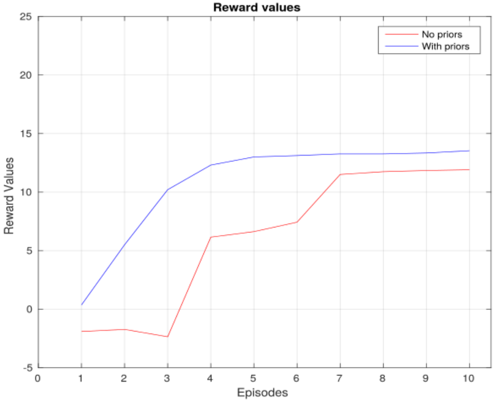


Fig: Reward comparison under damages

CONCLUSION

Conclusion

Contribution

- Black-DROPS successfully adapted for the grasping task involving high-dimensional state-action space of 11D
- Recovery from damages validated by both variants of the algorithm
- Modularising Black-DROPS for direct use with a real system through ROS controllers designed for implementing generated policies
- Vision module prepared from PCL, that can be used for any object detection purposes

Future Work

- Use of pressure/force sensors or visual information for grasp confirmation
- Extend the algorithm for use with minimal resets and adopt semi- episodic learning

THE END