# Policy gradient methods for controlling systems with discrete sensor information

**Matteo Gagliolo** MGAGLIOL@VUB.AC.BE
**Kevin Van Vaerenbergh**
**Abdel Rodríguez**
**Ann Nowé**
CoMo, VUB (Vrije Universiteit Brussel), Pleinlaan 2, 1050 Brussels, Belgium

**Stijn Goossens** STIJN.GOOSSENS@FMTC.BE
**Gregory Pinte**
**Wim Symens**
FMTC (Flanders' Mechatronics Technology Centre), Celestijnenlaan 300D, 3001 Heverlee, Belgium

## Abstract

In most existing motion control algorithms, a reference trajectory is tracked, based on a continuous measurement of the system's response. In many industrial applications, however, it is either not possible or too expensive to install sensors which measure the system's output over the complete stroke: instead, the motion can only be detected at certain discrete positions. The control objective in these systems is often not to track a complete trajectory accurately, but rather to achieve a given state at the sensor locations (e.g. to pass by the sensor at a given time, or with a given speed). Model-based control strategies are not suited for the control of these systems, due to the lack of sensor data. We are currently investigating the potential of a non-model-based learning strategy, Reinforcement Learning (RL), in dealing with this kind of discrete sensor information. Here, we describe experiments with a simple yet challenging system, where a single sensor detects the passage of a mass being pushed by a linear motor.

RL problems (Sutton & Barto, 1998) are a class of machine learning problems, where an agent must learn to interact with an unknown environment, using a "trial and error" approach. At a given timestep $t$, the agent may execute one of a set of *actions* $a \in \mathcal{A}$, possibly causing the environment to change its *state* $s \in \mathcal{S}$, and generate a (scalar) *reward* $r \in \mathbb{R}$. An agent is represented by a *policy*, mapping states to actions. The aim of a RL algorithm is to optimize the policy, maximizing the reward accumulated by the agent. Learning is organized in a sequence of *epochs*, each consisting of a sequence of interactions with the environment. Simply stated, RL consists in learning from a teacher (the environment) who cannot tell us *what* to do next (the optimal policy), but only *how good* we are doing so far (the reward signal). It therefore offers a suitable framework for the control of systems with discrete sensor information. The target state at the discrete sensors location can be incorporated in the reward signal, in order to favor the desired behavior.

The potential of different RL techniques is validated on a set-up consisting of a linear motor and a moving mass mounted on a straight horizontal guide (Figure 1). The position of the moving mass is monitored via a single discrete sensor, set along the guide, which fires at the passage of a small (1 cm) element attached to the mass. When the motor is activated, the mass is "punched" forward, and slides up to a certain position, depending on the duration and speed of the motor's stroke, and on the unknown friction among the mass and its guide.

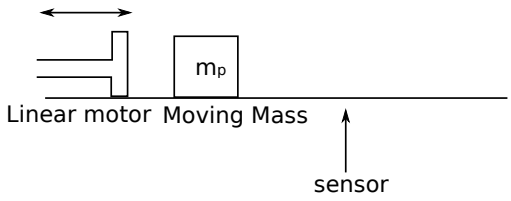


Figure 1: Sketch set-up, position measurement at one location

Two tasks are defined on this setup, with different objectives: a) let the mass pass the sensor at a predefined time

Appearing in *Proceedings of the 20th Machine Learning conference of Belgium and The Netherlands*. 

(time task); b) let the mass stop exactly in front of the sensor (position task). As the motor movement precedes the passing of the sensor, conventional feedback control methods cannot obviously be applied to solve these tasks. For simplicity, we only consider constant speed signals, with duration varying on a closed interval, such that an action consists of a single scalar, which we normalize in $[0,1]$, and an epoch consists of a single action, followed by a scalar reward. For each task, a corresponding reward function is implemented, favoring the desired behavior[1]. Fig. 2 reports samples of the two reward functions: note that the system is highly stochastic, and repeating the same action twice can lead to different rewards.

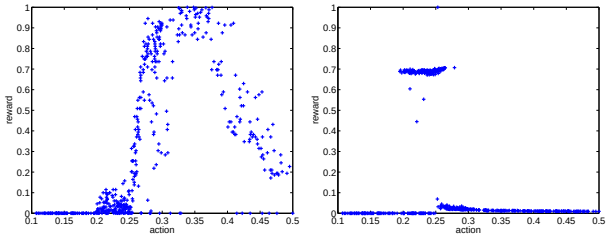


Figure 2: Reward functions for the two tasks, sampled for 500 randomly chosen actions on a subset of the unit interval.

Three *direct policy search* RL methods are evaluated on this setup, all representing the policy as a probability density function (pdf) over actions, which is updated offline, after each epoch: the policy gradient (PG) method (Peters & Schaal, 2006), where the actions are drawn from a Gaussian distribution; PG with parameter exploration (PGPE) (Sehnke et al., 2010), where the pdf is a mixture of Gaussians[2]; and a continuous learning automaton (CARLA) (Rodríguez et al., 2011), where the pdf over the actions is non-parametric.

While all three methods can successfully solve both tasks, CARLA displays faster convergence in both cases. Fig. 3, 4 report example runs on the two tasks. The position task turns out to be more difficult: this can easily be explained comparing the reward samples (Fig. 2). The best action for the position task, around 0.25, is a "needle in a haystack" compared to the time task, where the reward function changes more gradually around the optimal action.

[1]For the time task, given a target time $t_0$, reward is given as $r = \exp\{-c(t-t_0)^2\}$, where $c$ is a constant, and $t$ is the time at which the sensor starts firing, which is $\infty$ if the mass does not reach it. For the position task, the reward is given as the portion of time during which the sensor fires, over a constant time interval measured from the beginning of the motor's movement.

[2]This method is conceptually different from PG in that the pdf is not over actions, but over parameters of the policy, which are drawn at the beginning of an epoch. In this case, however, the action is a single parameter, and the epoch a single time step, so the two methods differ only for the pdf used (single Gaussian vs. mixture)

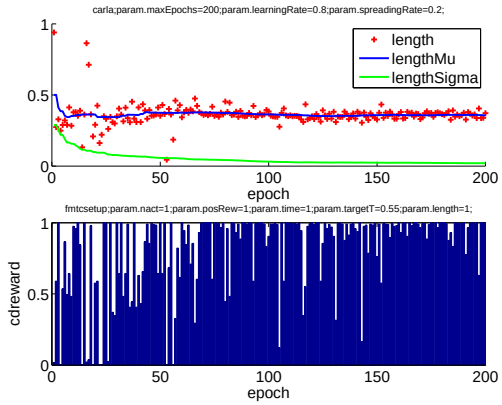


Figure 3: Time task, CARLA

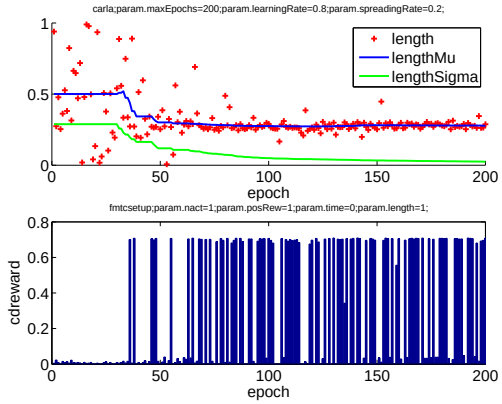


Figure 4: Position task, CARLA

**Acknowledgment.** This work has been carried out within the framework of the LeCoPro project (grant nr. 80032) of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

## References

Peters, J., & Schaal, S. (2006). Policy gradient methods for robotics. *Proceedings of the IEEE Intl. Conf. on Intelligent Robotics Systems (IROS)*.

Rodríguez, A., Grau, R., & Nowé, A. (2011). Continuous actions reinforcement learning automata. performance and convergence. *Intl. Conf. on Agents and Artificial Intelligence (ICAART)*.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., & Schmidhuber, J. (2010). Parameter-exploring policy gradients. *Neural Networks*, *23*, 551–559.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA, MIT Press.