Learning of Binocular Fixations using Anomaly Detection with Deep Reinforcement Learning

François Jean de La Bourdonnaye, Céline Teulière, Thierry Chateau, Jochen Triesch

Learning of Binocular Fixations using Anomaly Detection with Deep Reinforcement Learning

François de La Bourdonnaye,
Céline Teulière and Thierry Chateau
Université Clermont Auvergne, Institut Pascal
CNRS, UMR 6602
63178 Aubière, France

Jochen Triesch
Frankfurt Institute for Advanced Studies
Ruth-Moufang-Straße 1
60438 Frankfurt am Main, Germany

***Abstract*—Due to its ability to learn complex behaviors in high-dimensional state-action spaces, deep reinforcement learning algorithms have attracted much interest in the robotics community. For a practical reinforcement learning implementation on a robot, it has to be provided with an informative reward signal that makes it easy to discriminate the values of nearby states. To address this issue, prior information, e.g. in the form of a geometric model, or human supervision are often assumed. This paper proposes a method to learn binocular fixations without such prior information. Instead, it uses an informative reward requiring little supervised information. The reward computation is based on an anomaly detection mechanism which uses convolutional autoencoders. These detectors estimate in a weakly supervised way an object's pixellic position. This position estimate is affected by noise, which makes the reward signal noisy. We first show that this affects both the learning speed and the resulting policy. Then, we propose a method to partially remove the noise using regression on the detection change given sensor data. The binocular fixation task is learned in a simulated environment on an object training set with various shapes and colors. The learned policy is compared with another one learned with a highly informative and noiseless reward signal. The tests are carried out on the training set and on a test set of new objects. We observe similar performances, showing that the environment-encoding step can replace the prior information.**

I. Introduction

Autonomous learning of sensori-motor skills is of prime importance in uncertain and complex environments. Within this context, the field of developmental robotics is relevant since it replicates human infant abilities to learn with little supervision. E.g. children learn skills such as fixating and grasping objects or walking mostly autonomously without much external supervision. For this setting, reinforcement learning (RL) is a particularly appealing framework.

In the past few years, RL applications have benefited from progress in deep learning and have become increasingly complex in terms of state space size. For example, [1] successfully learns pixel-to-move policies on different Atari games. Pixel-to-torque policies have also been learned in several manipulation robotics applications [2], [3], [4]. Novel RL approaches were demonstrated on high-dimensional simulated tasks [5], [6], [7].

One of the limitations of these applications is the use of external supervision for the reward function. Indeed, to ensure a practical deep reinforcement learning application, reward signals must properly discriminate the values of certain close states and must not be too noisy. For this, external supervision is often used. In Atari games, the environment provides the score which is an informative reward that does not apply to the real world setting. In manipulation robotics, a reward function is often defined based on a distance measure between the end-effector current position and a target pose. This generally requires a geometric model and object pose information. Moreover, when the desired policy has to generalize over several target poses, finding the target position information is often tedious. The need to dispense with human supervision is also expressed in [8]. The authors choose a simple way to define a goal in various situations and apply their idea along with model predictive control in a box-pushing task. It consists of choosing some pixels and their target positions. However, defining those target locations requires supervision.

The aim of this paper is to learn an object fixation task using neither prior information about the environment (object poses) nor geometric models or camera parameters. More precisely, the task consists of fixating a random object located at a random position on a table with both cameras. This learned skill could be reused later to simplify manipulation tasks such as reaching or grasping. Humans first look at an object and then grasp it and this strategy reduces the complexity of the reaching problem. The policy is learned using the deep deterministic policy gradient algorithm [5], mapping pixels and camera coordinates to camera movements. To design a reward function requiring as little external information as possible, we propose to use a weakly-supervised anomaly detection mechanism, which can also be related to a maximal-entropy seeking mechanism. In our approach, the object is viewed as an anomaly with respect to the agent's knowledge about the environment. The goal is to locate it in the image and then bring it to the image center.

Our object detection approach belongs to the class of semi-supervised anomaly detection methods. It is similar to [9] where learning to detect network intrusions and breast cancers is proposed. To that aim, a multi-layer perceptron is used as an autoencoder to reconstruct inputs labelled as "with anomalies" or "anomaly free". A positive aspect similar to ours is that it allows to locate the anomaly. However, the applications are rather low-dimensional since no more than 50 neural network

inputs are used in the experiments. [10] uses support vector machine methods to learn how to detect outliers in digit images. For each kind of digit, a support vector machine is trained to model a density. Abnormal images are detected when they are incompatible with the trained density. Similar to [9], the problem is rather low-dimensional since it involves 256 binary dimensions.

These approaches are labelled as semi-supervised methods because they use very few labels while accomplishing unsupervised work such as density estimation or self-reconstruction. For instance, [10] uses class labels for each kind of digit while learning densities, [9] uses the labels "with anomalies" and "anomaly free" while learning self-reconstruction. In our work, an autoencoder is first learnt without any object in the environment. Then, if an object is added to the environment, the autoencoder reconstruction error map allows to estimate the object pixellic position. We choose to refer to our work as "weakly supervised" to highlight the fact that the supervised information is both easy to find and limited. Indeed, the proposed approach does not use any human supervision with two minor exceptions. First, the image center is computed and is required for the reward function. However, this computation can be viewed as universal. Second, the autoencoder training database is labelled as "without object" and the learning phase is always achieved with an object present in the scene, which is not particularly tedious. After the object detection step, the reward is computed by calculating the distance between the estimated object position and the image center.

Another contribution of the proposed work is the handling of reward noise. Indeed, due to impulse noise present in the object position estimation, learning has to deal with very noisy rewards, which is not the case in most deep reinforcement learning applications. Previously, [11] used a moving average filter on the temporal reward signal with good results at removing impulse and Gaussian noise in a low-dimensional gridworld problem. This method can be applied to high-dimensional RL problems but it would modify noiseless rewards. More recently, [12] studied the influence of noise on the state-action value function overestimation. They developed a method to cancel this effect, penalizing unlikely actions in the update given prior knowledge. This method cannot be integrated in our work since we want as little prior information as possible. In our method, we choose to apply a learning approach to remove the reward impulse noise. First, we show that this reward noise effectively reduces learning speed. Second, we propose a solution by learning a mapping between movement amplitudes and object detection changes. This is achieved by a generic regression algorithm. The method for noise removal can be generalized to other learning applications with rewards affected by impulse noise.

The last interesting feature of the proposed method concerns generalization. The learned sensori-motor mapping for object fixation has to apply to any object. To achieve this, the system learns on a training set of objects with various shapes and colors and its generalization capability is demonstrated on a test set of new objects.

The remainder of the paper is organized as follows. Section II reviews the theoretical tools used in the paper and presents the weakly supervised reward computation framework. Section III describes the experiments made to test the reward computation framework and presents the results. Section IV discusses the work from a broader perspective.

II. Methods

A. Background

1) Reinforcement learning:
A reinforcement learning framework is usually defined on a Markov decision process $< S, A, R, T >$ where:

- S is the set of states
- A is the set of actions
- T is the transition model ($T : S \times A \to S$)
- R is the reward function ($R : S \times A \times S \to \mathbb{R}$)

The goal of a reinforcement learning agent is to optimize a criterion based on future rewards. Here, a sum of discounted future rewards is used as optimization criterion:

$$J = \sum_{k=0}^{\infty} r_k \gamma^k, \tag{1}$$

where $\gamma \in [0, 1]$ is a discount factor.

To optimize the criterion, we train a deterministic policy $\pi : S \to A$. In the paper, the state-action value function is used:

$$Q_\pi(s, a) = E_\pi \left[\sum_{k=0}^{\infty} r_k \gamma^k \middle| s, a \right]. \tag{2}$$

2) Deep reinforcement learning:
Reinforcement learning suffers from with the "curse of dimensionality"[13]. Since 2010's, with the emergence of the deep learning field and the arrival of better computers and GPUs, the usual value functions Q, A or V, the policies and dynamics are more and more frequently approximated by deep neural networks. Using deep approximators allows to represent high-dimensional functions by automatically extracting a hierarchy of features. Furthermore, the state space of the RL tasks is more and more often high-dimensional. The first so-called deep reinforcement learning application learned on raw pixels without using hand-designed features [14] uses a combination of batch Q learning with deep autoencoders to reduce the dimensionality of the state space. The deep Q-network (DQN) [1] beats human experts in many of the Atari games. It uses a convolutional neural network to approximate the Q function. A stable and reliable learning is obtained by using two tricks. First, in order to have a stable learning, the Q update uses a target network. It is a Q-network whose weights are frozen for a given number of iterations. Second, in order to have a mini-batch of i.i.d samples, they are uniformly chosen from a large memory buffer.

3) Deep deterministic policy gradient algorithm:

Even though the DQN achieves impressive performances in Atari games, it cannot be applied to tasks involving a continuous action space. The deep deterministic policy gradient algorithm (DDPG) [5] leverages this limitation by combining the deterministic policy gradient algorithm [15] and the DQN [1]. It is an "actor-critic" algorithm updating the critic Q_ϕ with parameters ϕ and the deterministic policy π_θ with parameters θ according to the equations below:

Let T_b be a batch of transitions:

$< s_i, a_i, r_i, s_i^{'} >_{i\in\{1,...,N_\mathrm{b}\}} \in S \times A \times \mathbb{R} \times S$, with N_b being the batch size.

The targets of the Q_ϕ neural network are computed using a TD(0) update (with a learning rate equal to 1):

$$\forall i \in \{1, ..., N_\mathrm{b}\}, y_i = r_i + \gamma Q_\mathrm{T}(s_i^{'}, \pi_\theta(s_i^{'})), \tag{3}$$

with Q_T being a target network which copies Q_ϕ every K iterations. The Q_ϕ network updates its weights by minimizing the least square error $\frac{1}{2N_\mathrm{b}} \sum_{i=1}^{N_\mathrm{b}} (y_i - Q_\phi(s_i, a_i))^2$. Using the Q_ϕ network and the fact that the policy is deterministic, the following policy gradient is derived:

$$\frac{\partial Q_\phi}{\partial \theta} \simeq \frac{1}{N_b} \sum_{i=1}^{N_b} \frac{\partial Q_\phi(s_i, \pi_\theta(s_i))}{\partial a} \frac{\partial \pi_\theta(s_i)}{\partial \theta}. \tag{4}$$

This update makes the policy select the actions that maximize the Q function at the batch states. Note that in [5], a target network is used for both the policy and the Q function. In our work, it is used only for the Q function.

In addition to this algorithm, we use the inverting gradient procedure of [16] to bound the actions. This method downscales the gradient when the action computed by the policy approaches its limit. When it exceeds its limit, the gradient is inverted. This mechanism prevents the actions from being too large.

B. Fixation Task definition

1) Task:

The fixation on an object is achieved when the object is at the center of both cameras (see Figure 1).

Mathematically, the problem is modelled as a Markov decision process where:

- $S = (I^{\mathrm{left}}, I^{\mathrm{right}}, q)$, the set of states involves the two RGB left and right images (I^{left} and I^{right}) and the camera coordinates q (three continuous scalars). The camera coordinates $q = (J_1, J_2, J_3)$ are represented by two pan angles for each camera and one joint tilt angle (see Figure 2).
  With a perfect object detector, the object pixellic position could be used directly as a state instead of images. However, impulse noise affecting the object detection makes this unpractical. Learning a direct mapping from images to actions solves this issue and avoids the need for any object detector during the exploitation.
- A, the set of actions comprises the three variations of camera coordinates (three continuous scalars)
  $A = \Delta q = (\Delta J_1, \Delta J_2, \Delta J_3)$

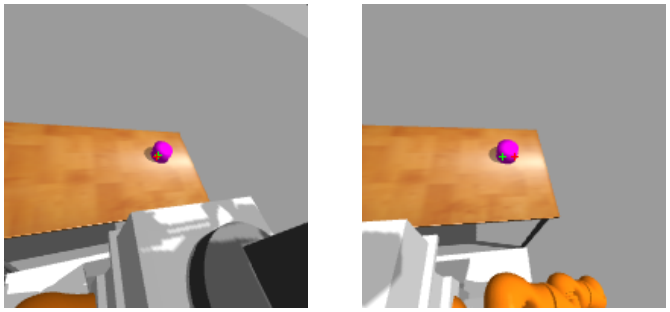

Fig. 1. Binocular fixation achieved on a purple cylinder, the red cross stands for the image center, the green cross is the estimated object pixellic position according to the method described in II-C1b

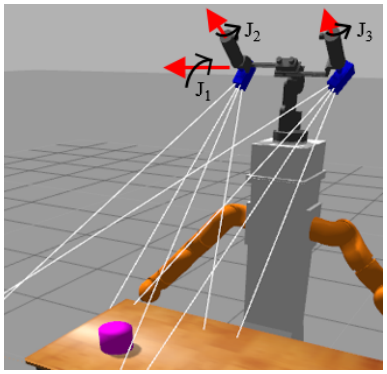

Fig. 2. Representation of the two-cameras system with its three DOFs

2) Neural network structures:

The structures of the Q function and the policy are presented in Figure 3. They have been chosen to optimize learning performance (improvement of policy, loss of the Q network).

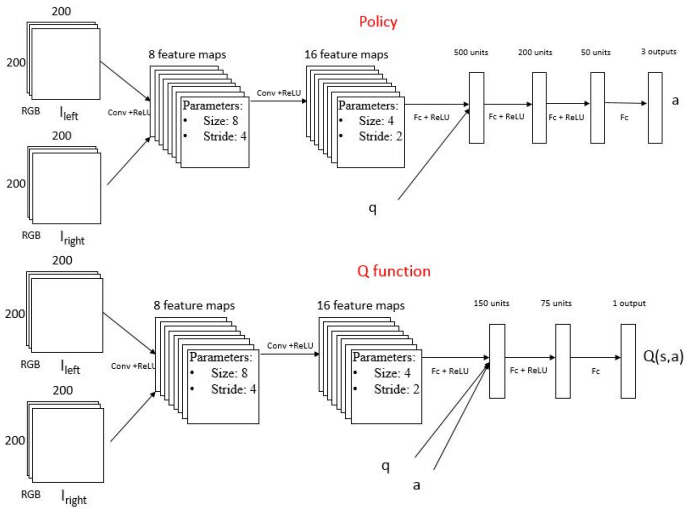

Fig. 3. Structure of the policy and the Q function

C. Reward computation framework

The following section describes how the weakly supervised reward is computed.

1) General framework for the reward computation:

The reward computation involves three steps:

- A pre-training step in which the agent learns to reconstruct the environment without object
- An object detection step
- The reward computation step itself

In the following paragraphs, the superscript c represents the camera left or right.

a) Pre-training step:

For each camera $c = \text{left or right}$, 10000 camera configurations are generated leading to two 10000-sized image databases D^{left} and D^{right}. The set of camera configurations covers a regular grid of configurations between the joint limits (arbitrarily fixed to keep the table inside the field of view).

The images are converted from RGB 200×200 format to grayscale 50×50 images.

Then, the autoencoder A_{ψ^c} is trained on D^c with the help of the Adam solver [17].

b) Object detection:

The object detection step takes into account the reconstruction error map. The rationale behind the method is that objects are badly reconstructed since autoencoders are not trained on them. It makes the reconstruction error localized at the object position. This feature allows to estimate the object pixellic position using a kernel density estimator.

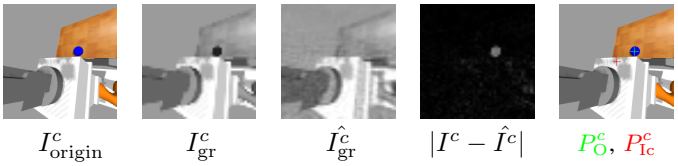

Fig. 4. Object detection computation scheme

Figure 4 shows the different steps of the reward computation:

- The image is downsampled and converted into a grayscale one $I^c_{\text{origin}} \rightarrow I^c_{\text{gr}}$.
- The image is reconstructed using the learned autoencoder $I^c_{\text{gr}} \rightarrow \hat{I^c_{\text{gr}}} = A_{\psi^c}(I^c_{\text{gr}})$.
- The error map is computed $|I^c - \hat{I}^c|$.
- From the error map, the N points $\{P(i)\}_{i\in\{1,\dots,N\}} = \{(x(i), y(i))\}_{i\in\{1,\dots,N\}}$ with the highest intensity are extracted. $\{I(i)\}_{i\in\{1,\dots,N\}}$ is the set of corresponding luminances.

  From these points, a probability distribution $\{p(i)\}_{i\in\{1,\dots,N\}}$ is computed using a kernel density estimator with a Gaussian kernel of zero mean and unit variance:

$$\forall i \in \{1, ..., N\}, p(i) = \frac{1}{N}\sum_{j=1}^{N} I(j)K(P_i - P_j), \quad (5)$$

with $K(P_i - P_j) = \frac{1}{2\pi}\exp^{-0.5||P_i - P_j||_2^2}$.

The estimated object pixellic position P^c_O is at the maximal probability:

$$P^c_O = P_k, \quad (6)$$

where $k = \arg\max_i(p(i))$. N is set to 150 in the experiments.

c) Reward computation:

The reward is a decreasing function of the Euclidean distance $||P^c_{\text{O}} - P^c_{\text{Ic}}||_2$. We propose to use an affine function clamping the values between -1 and 1 for each camera:

$$r^c = 2 \times \left(\frac{d_{\max} - ||P^c_{\text{O}} - P^c_{\text{Ic}}||_2}{d_{\max}} - \frac{1}{2}\right), \quad (7)$$

where $d_{\max}$ is the maximal Euclidean distance between the object pixellic position and the image center.

$$r = r^{\text{left}} + r^{\text{right}} \quad (8)$$

As aforementioned, this method allows to extract a reward signal in a weakly supervised way. Moreover, this reward is informative in the sense it can discriminate the values of nearby states.

2) Autoencoder structure:

The autoencoder structure is presented in Figure 5. The encoder is composed of a convolutional layer and two fully connected layers. The decoder is composed of two fully connected layers. The last fully connected layer of the encoder and the first fully connected layer of the decoder have transposed weights.

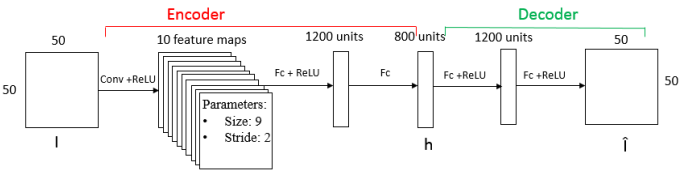

Fig. 5. The autoencoder structure

The choice of the architecture (number of layers, type of layers, number of neurons in the third hidden layer) is made experimentally by trying to get good performances while ensuring fast learning.

3) Reward noise removal method:

The learned autoencoders do not reconstruct perfectly the images even without object in the scene. Indeed, some high-frequency areas such as table legs are difficult to reconstruct. Consequently, high-frequency areas are sometimes detected, producing impulse noise in the reward function (cf Figure 6). Experiments in III-C show that this has a strong influence on learning even if the impulse noise only concerns about 3% of the samples. We choose to model the function $\Delta\text{d} = f(||\Delta q||_2)$ to detect an abnormal object detection variation in function of the movement amplitude. Δd is the Euclidean distance in the pixel space between two successive object detections.

This function can be approximated by any regression method including neural networks, Gaussian processes or support-vector machines. Here, a Gaussian process with a squared exponential interaction function is used and trained with 1000 transitions. Using it allows to remove transitions whose difference (in pixels) between the real detection variation and the estimated detection variation is above a threshold.

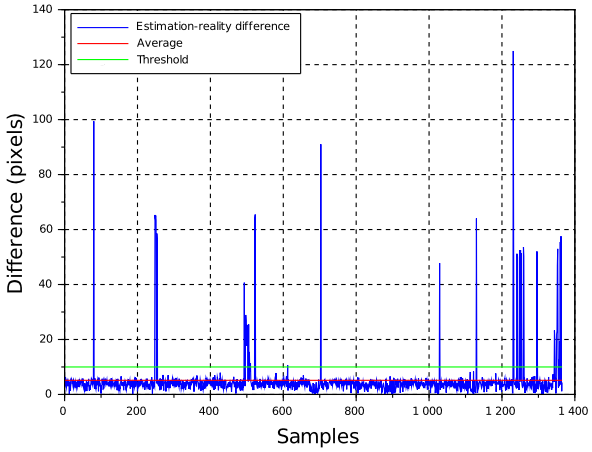

Fig. 6. Evolution of the error in object motion estimation.

Figure 6 illustrates the partial noise removal results and reveals the impulse nature of the noise. The threshold has been set to 10, removing around 3% of the samples.

III. Results

A. Experimental environment

The experiments are carried out in a simulated environment using the Gazebo simulator jointly with the ROS middleware. A two-cameras system is mounted on a robotic platform (cf Figure 2). A table from the Gazebo database is placed below the cameras and an object is put on it. The experiments involve two object sets. The training and the test sets can be seen in Figure 7. The databases consist of objects from the Gazebo models and hand-designed objects. In order to potentially generalize the fixating skill to new objects, the training set objects were designed with diverse colors and shapes (cylinder, parallelepiped, and sphere). Every test object has a new shape compared with training objects and a new color (turquoise) appears in the test set. The goal is to check if the learned policy can achieve fixations on objects whose color or shape is not present in the training set.

B. Implementation details

For all the neural network algorithms, we use the caffe library [18]. A GPU (nvidia GeForce GTX Titan X) is used for the experiments. With this hardware set-up, 10 000 learning iterations of DDPG correspond to about 40 minutes of training.

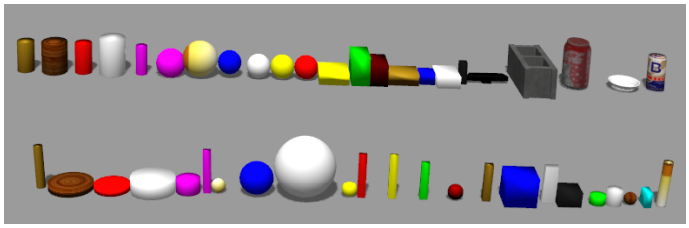

Fig. 7. Training and test sets

Batch normalization [19] is used to normalize layer inputs and avoid burdensome pre-processing steps. The solver Adam is used to train the Q function. The learning rate of the policy network is hand-tuned.

Hyperparameter values mentioned in Algorithms 1 and 2 are listed in Table I.

Hyperparameters	ϵ	N_{tot}	N_{eps}	N_{b}	th	N_{gp}
Training	0.02	150000	35	16	10	1000
Test		2000	35			

TABLE I
HYPERPARAMETER VALUES

C. Training

Algorithm 1 presents the binocular fixation learning procedure which uses an episodic set-up. In order to avoid the problem of correlated data, the object and its position regularly change according to a uniform probability distribution. Note that instead of using the Ornstein-Uhlenbeck process for the exploration as proposed in [5], a constant zero-mean Gaussian noise is used. With an informative reward function, this simple exploration setting is sufficient for learning the fixation task. The number of iterations of the algorithm is set to 150000 because the policy does not improve anymore afterward.

In order to ensure a fast learning, two tricks are implemented in the training procedure:

- We add a term to the reward penalizing both divergence and a too strong convergence. More precisely, if $d_{\text{pan}} = J_3 - J_2$ is the angular difference between the pan left and right angles, then the additional reward term r_{pan} is computed as an affine function of d_{pan} when $d_{\text{pan}} > 0$ (eye divergence) or $d_{\text{pan}} < -0.4$ (strong eye convergence):

$$r_{\text{pan}} = \begin{cases} \frac{-d_{\text{pan}}}{3}, & \text{if } d_{pan} > 0 \\ \frac{d_{\text{pan}}+0.4}{3}, & \text{if } d_{pan} < -0.4 \\ 0.1, & \text{otherwise.} \end{cases} \quad (9)$$

- At each episode end, the camera system is set to the same initial position.

The objective of the experiments on training is twofold. First, we want to show how the impulse noise affects learning. Second, we want to demonstrate that learning with the filtered weakly supervised reward gives similar training performance

Algorithm 1 Training procedure

Parameters:

The initial position s_0
The Gaussian noise variance ϵ
The total number of iterations N_{tot}
The number of iterations per episode N_{eps}
The number of transitions per batch N_{b}
The removal threshold th
The number of samples required to train Gaussian process N_{gp}

Inputs:

The training set D_{train}

Outputs: Q_ϕ, π_θ

Steps:

Set $t \leftarrow 0$
Initialize the transition circular buffer T_{buf}
Set $cond \leftarrow (t < N_{\text{gp}})$ or $((|\Delta \text{d} - f(||\Delta q||_2)| < th)$ and $(t > N_{\text{gp}}))$
while $t < N_{\text{tot}}$ **do**
    Choose a random object in D_{train} and place it randomly
    Go to the initial position s_0
    Set $t_{\text{eps}} \leftarrow 0$
    while $t_{\text{eps}} < N_{\text{eps}}$ **do**
        Apply $a_t = \pi_\theta(s_t) + \mathcal{N}(0, \epsilon)$
        Observe s_{t+1}
        Compute r_t and $|\Delta \text{d}_t|$
        if $cond$ **then**
            Add $< s_t, a_t, r_t, s_{t+1} >$ to T_{buf}
        Pick randomly N_{b} transitions from T_{buf}
        Update Q_ϕ and π_θ using DDPG [5]
        if $t = N_{\text{gp}}$ **then** Train $\Delta \text{d} = f(||\Delta q||_2)$
        $t \leftarrow t + 1$
        $t_{\text{eps}} \leftarrow t_{\text{eps}} + 1$
    Withdraw the object

as with a supervised reward. The supervised reward is obtained by computing the Euclidean distance between the projection of the object center of gravity and the image center $||P^C_{\text{pr}} - P^C_{\text{Ic}}||_2$. This function is also affine with the same parameters as for the weakly supervised one:

$$r^C_{\text{sup}} = 2 \times \left(\frac{d_{\max} - ||P^C_{\text{pr}} - P^C_{\text{Ic}}||_2}{d_{\max}} - \frac{1}{2} \right), \tag{10}$$

$$r_{\text{sup}} = r^{\text{left}}_{\text{sup}} + r^{\text{right}}_{\text{sup}}. \tag{11}$$

It requires to compute the camera extrinsic parameters at each iteration and to do a calibration before learning.

To achieve the objectives, learning improvements with the supervised, noisy weakly supervised and filtered weakly supervised rewards are compared. The fixation error is tracked over time. We average three experiments for each case. It is done by using the random variable:

$$e_{\text{p}}(t) = \frac{||P^{\text{left}}_{\text{p}}(t) - P^{\text{left}}_{\text{Ic}}||_2 + ||P^{\text{right}}_{\text{p}}(t) - P^{\text{right}}_{\text{Ic}}||_2}{2}. \tag{12}$$

It represents the average (over the left and right images) Euclidean distance between the image center and the projection of the object center of gravity.

The results are shown in Figure 8 (the curves are filtered using an exponential smoothing for better readibility). The vertical axis of the plot represents the aforementioned $e_{\text{p}}(t)$ variable whereas the horizontal one stands for the time steps. The curve $e^{\text{s}}_{\text{p}}(t)$ represents $e_{\text{p}}(t)$ with the use of the supervised reward, $e^{\text{w}}_{\text{p}}(t)$ with the noisy weakly supervised reward and $e^{\text{wf}}_{\text{p}}(t)$ with the filtered weakly supervised reward.

The curve $e^{\text{w}}_{\text{p}}(t)$ presents the worst learning performances. It shows that the noise affects learning.

The curves $e^{\text{s}}_{\text{p}}(t)$ and $e^{\text{wf}}_{\text{p}}(t)$ have similar shapes. We observe that $e^{\text{s}}_{\text{p}}(t)$ decreases a little more quickly. This small difference is due to the remaining noise present in the samples.

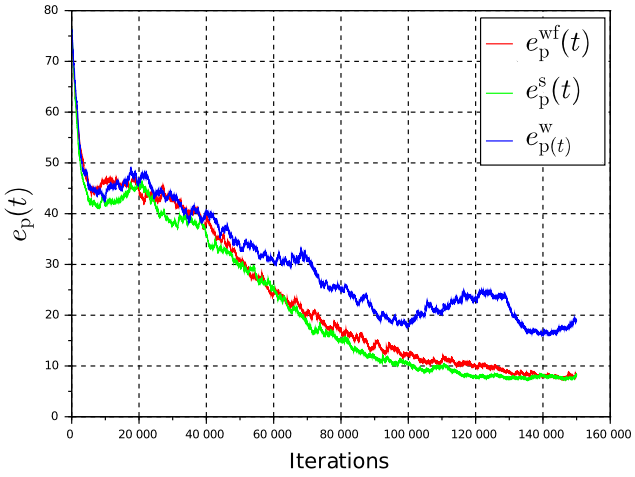

Fig. 8. Binocular fixation position error over time

D. Test

The objective of the policy test can be divided into three sub-goals:

Firstly, we want to compare the performance of the learned policy on the test set with that of the training set. This aims at checking if there is no overfitting. Secondly, we want to compare the policies learned with the noisy reward and with the filtered one. The aim is to show that not only does the noise alter learning, but the performance of the resulting policy is also worse. Thirdly, we want to compare it with the policy learned with an informative and noiseless reward signal. The purpose is to check if the weakly supervised reward allows to learn the same kind of behavior as for the supervised reward.

Algorithm 2 Test procedure

Parameters:

The total number of episodes N_{tot}

The number of iterations per episode N_{eps}

Inputs:

The policy π_θ

The object set $D = D_{\text{test}}$ or D_{train}

Output: The set of final fixation errors D_{p}

Steps:

Set $t \leftarrow 0$
while $t < N_{\text{tot}}$ **do**
    Choose a random object in D and place it randomly
    Go to the initial position s_0
    Set $t_{\text{eps}} \leftarrow 0$
    while $t_{\text{eps}} < N_{\text{eps}}$ **do**
        Apply $a_t = \pi_\theta(s_t)$
        Observe s_{t+1}
        $t_{\text{eps}} \leftarrow t_{\text{eps}} + 1$
    $t \leftarrow t + 1$
    Compute the fixation error $e_{\text{p}}(t_{\text{eps}})$ and append it to D_{p}
    Withdraw the object

	Training set			Test set		
Statistics (pixels)	r_{w}	r_{wf}	r_{s}	r_{w}	r_{wf}	r_{s}
$mean(e_{\text{p}})$	14.11	4.95	4.80	17.10	5.92	6.81
$median(e_{\text{p}})$	8.57	4.35	4.28	11.20	4.85	5.39
$std(e_{\text{p}})$	14.66	5.31	3.02	17.77	6.63	8.42

TABLE II
PERFORMANCES OF THE LEARNED POLICIES

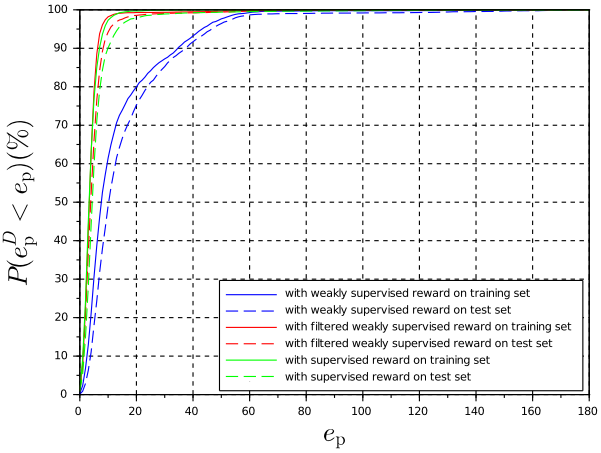

Fig. 9. Cumulative distribution function of the fixation error (in %)

Algorithm 2 describes the test procedure. We choose to evaluate at the end of each episode the random variable $e_{\text{p}}(t_{\text{eps}})$. Statistics (mean, median and standard deviations) of these random variables are computed for 2000 test episodes. The value of 2000 was set to ensure that the computed statistics are meaningful. Furthermore, the results are averaged over three policies for each reward case. Finally, a cumulative percent curve is presented to more precisely characterize the distribution of $e_{\text{p}}(t_{\text{eps}})$.

The results are summarized in Table II. The columns r_{w}, r_{wf} and r_{s} respectively stand for the case of the noisy weakly supervised, filtered weakly supervised and supervised rewards, respectively. Figure 9 gives a more complete view of the distribution of the fixation error. The vertical axis $P(e_{\text{p}}^D < e_{\text{p}})$ represents the percentage of tested samples whose fixation error is smaller than the values on the horizontal axis.

Several remarks can be made:

- The results of the policy learned with the noisy weakly supervised reward are the worst.
- The results of the policies learned with the supervised and filtered weakly supervised rewards do not exhibit a big difference in both the training and the test set. The differences that we observe are due to the inter-experimental variance and the fact we average the results over 3 trials.
- The errors are higher for the test set compared with those on the training set. However, the difference is not high. As a consequence, there is no strong overfitting.

IV. Discussion

We have presented an approach for the learning of binocular fixations based on anomaly detection with deep reinforcement learning that requires only minimal supervision. Our experiments show that the learned policy generalizes well to new objects. We have shown that a form of impulse noise in the object detection affects both the learning speed and the quality of the resulting policy. We have proposed a noise removal method to deal with this challenge that greatly improves the quality of learning. Moreover, our experiments show that learning with the filtered weakly supervised or the supervised reward leads to similar performance both during training and testing. However, learning is a bit slower in the case of the weakly supervised reward because of the residual impulse noise. Overall, our study shows that the proposed reward function, while using hardly any supervised information, is well suited for learning the binocular fixation task. This indicates that prior information that is usually used in other systems [5] might be replaced by an autoencoder training step coupled with an anomaly detection mechanism.

The reward computation method still has several limitations, however:

- The environment cannot vary during learning. Otherwise, the autoencoder must be adapted (this task is difficult since during learning, the object is in the environment).
- The method works in a rather simple setting and has not been adapted yet to a cluttered environment with many

objects.

In future work we would like to make the autoencoder adaptive to increase the system robustness. Furthermore, we would like to validate the method on a physical robot learning in the real world. Finally, we wish to adapt our method to other types of task such as object reaching or grasping.

ACKNOWLEDGMENT

This work is sponsored by the French government research program "Investissements d'avenir" through the IMobS3 Laboratory of Excellence (ANR-10-LABX-16-01), by the European Union through the program Regional competitiveness and employment (ERDF Auvergne region), and by the Auvergne region. JT acknowledges support from the Quandt foundation.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu , D. Silver, A. Rusu Andrei, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski , S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg , and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end Training of Deep Visuomotor Policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.

[3] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation." in *ICRA*, 2016, pp. 378–383.

[4] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *ICRA*, 2016, pp. 512–519.

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning." *CoRR*, vol. abs/1509.02971, 2015.

[6] N. Heess, G. Wayne, D. Silver, T. P. Lillicrap, T. Erez, and Y. Tassa, "Learning Continuous Control Policies by Stochastic Value Gradients." in *NIPS*, 2015, pp. 2944–2952.

[7] S. Gu, T. P. Lillicrap, I. Sutskever, and S. Levine, "Continuous Deep Q-Learning with Model-based Acceleration." in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 48, 2016, pp. 2829–2838.

[8] C. Finn and S. Levine, "Deep Visual Foresight for Planning Robot Motion," *ArXiv*, 2016.

[9] S. Hawkins, H. He, G. J. Williams, and R. A. Baxter, "Outlier Detection Using Replicator Neural Networks." in *DaWaK*, ser. Lecture Notes in Computer Science, vol. 2454, 2002, pp. 170–180.

[10] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support Vector Method for Novelty Detection." in *NIPS*, 1999, pp. 582–588.

[11] A. Moreno, J. D. Martin, E. Soria, R. Magdalena, and M. Martinez, "Noisy Reinforcements in reinforcement learning: some case studies based on gridworlds," in *WSEAS*, 2006, pp. 296–300.

[12] R. Fox, A. Pakman, and N. Tishby, "Taming the Noise in Reinforcement Learning via Soft Updates," in *UAI*, 2016.

[13] R. E. Bellman, *Adaptive Control Processes: A Guided Tour*. MIT Press, 1961.

[14] S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *IJCNN*, 2010, pp. 1–8.

[15] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms," in *ICML*, Beijing, China, 2014.

[16] M. J. Hausknecht and P. Stone, "Deep Reinforcement Learning in Parameterized Action Space." *CoRR*, vol. abs/1511.04143, 2015.

[17] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *ICLR*, 2015.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," *arXiv*, 2014.

[19] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *ICML*, 2015, pp. 448–456.