Adaptive optics control with multi-agent model-free reinforcement learning

B. Pou, F. Ferreira, E. Quinones, D. Gratadour, M. Martin

Research Article Vol. 30, No. 2 / 17 Jan 2022 / *Optics Express* 2991

Optics EXPRESS

Adaptive optics control with multi-agent model-free reinforcement learning

B. Pou,[1,2,*] F. Ferreira,[3] E. Quinones,[1] D. Gratadour,[3,4] and M. Martin[2]

[1] *Barcelona Supercomputing Center (BSC), C. Jordi Girona, 29, 08034 Barcelona, Spain*
[2] *Computer Science Department, Universitat Politècnica de Catalunya (UPC), C. Jordi Girona, 31, 08034 Barcelona, Spain*
[3] *LESIA, Observatoire de Paris, Universite PSL, CNRS, Sorbonne Universite, Univ. Paris Diderot, Sorbonne Paris Cite, 5 place Jules Janssen, 92195 Meudon, France*
[4] *Research School of Astronomy and Astrophysics, Australian National University, Canberra, ACT 2611, Australia*
[*] *bartomeu.poumulet@bsc.es*

Abstract: We present a novel formulation of closed-loop adaptive optics (AO) control as a multi-agent reinforcement learning (MARL) problem in which the controller is able to learn a non-linear policy and does not need *a priori* information on the dynamics of the atmosphere. We identify the different challenges of applying a reinforcement learning (RL) method to AO and, to solve them, propose the combination of model-free MARL for control with an autoencoder neural network to mitigate the effect of noise. Moreover, we extend current existing methods of error budget analysis to include a RL controller. The experimental results for an 8m telescope equipped with a 40x40 Shack-Hartmann system show a significant increase in performance over the integrator baseline and comparable performance to a model-based predictive approach, a linear quadratic Gaussian controller with perfect knowledge of atmospheric conditions. Finally, the error budget analysis provides evidence that the RL controller is partially compensating for bandwidth error and is helping to mitigate the propagation of aliasing.

1. Introduction

Closed-loop Adaptive Optics (AO) systems are a fundamental component of large telescopes to correct dynamically evolving wavefront aberrations introduced by the atmosphere in real-time. AO systems are generally composed of one or several wavefront sensor(s) (WFS), based on e.g. the Shack-Hartmann concept (SH-WFS), a real-time controller (RTC), and one or several deformable mirror(s) (DM). At each iteration of the AO loop, the wavefront shape is captured by the WFS and reconstructed by the RTC to issue commands to the DM's actuators in order to compensate for the incoming perturbations.

The RTC is the key element in the loop that computes the appropriate commands to the DM. The classical AO controller relies on a linear relationship between measurements from the WFS and DM commands and an integrator for which a gain factor is used to mitigate errors introduced by e.g. intrinsic loop delay and temporal sampling. More advanced approaches have been developed under the form of model-based predictive controllers which predict future wavefront distortions and issue the appropriate commands to solve them. This is the case of linear quadratic Gaussian (LQG) controllers, first introduced in [1], which involve a linear dynamics model to describe the evolution of the system, a quadratic loss function to be optimised and Kalman filters to process noisy inputs. The required dynamics model is typically built via different *a priori* assumptions on the atmosphere (e.g. frozen flow) and its coefficients are usually obtained offline, through identification methods based on Machine Learning (ML) techniques to process AO telemetry data [2,3]. The work from several authors has improved upon [1], some examples

#444099 https://doi.org/10.1364/OE.444099
Journal © 2022 Received 27 Sep 2021; revised 24 Dec 2021; accepted 26 Dec 2021; published 14 Jan 2022

being its extension to Multi-Conjugate Adaptive Optics [4,5], its detailed analysis from a control theory point of view [6] and its possible real-time implementation with distributed Kalman filters [7]. Overall, LQG is a linear control approach that depends on properly capturing the atmosphere dynamics. As a result, incorrect *a priori* assumptions as well as non-linear effects present in the system impact its performance.

An alternative, explored by a number of authors, is to build the solution purely from data and without prior information. For instance, [8] presents a predictive method that uses Empirical Orthogonal Functions to predict future wavefronts as a linear combination from previous wavefronts estimates and issue the necessary commands to correct for them. Non-linear data-driven ML methods with deep neural networks have also been proposed for AO. Those neural network methods have been used either as predictive tools or as non-linear reconstructors of wavefronts. In their early work, [9] used a single hidden layer perceptron to predict future WFS measurements from previous ones and [10] used a multi-layer perceptron in a human vision setting to reconstruct the wavefront phase from the WFS measurements as a combination of Zernike polynomials.

Recently, more advanced neural network architectures have been used, such as convolutional neural networks (CNN) or recurrent neural networks (RNN), which exploit spatial and temporal patterns, respectively, for either reconstruction or prediction [11–14]. However, most of these neural network studies showed the quality of prediction/reconstruction but did not extend the method to real-time control in a closed-loop setting except for [13,14]. The work of [13] presented a CNN used to reconstruct the wavefront phase from pyramid WFS images [15] and showed an increase in closed-loop performance compared to a traditional controller when the non-linearities were substantial. [14] presented a Supervised Learning method using either RNN or CNN to predict future wavefront slopes that were then used to obtain the desired commands. Their novel idea was to combine the predictive model with a Generative Adversarial Network, which provided means to compute a regularising auxiliary loss. This regularisation allowed both architectures to work in closed-loop as opposed to similar previous work.

Despite their successes, the neural network approaches mentioned above are Supervised ML methods that require datasets with pairs (input, correct output) and are trained offline. This correct output can only be obtained in a controlled setting, for instance in a simulation, and the result may be biased towards wrong assumptions that such a controlled setting makes.

In this work, we focus on newly developed ML techniques to develop an AO controller that does not require neither a previously obtained dataset nor prior assumptions and can be trained online. Specifically, we use Reinforcement Learning (RL) methods, which involve learning a controller via trial and error that maximises a reward signal. Deep RL, the combination of RL with deep neural networks which provide the non-linearity, has already been used successfully in various real-world control applications such as end-to-end learning for robotic tasks [16], resource management in systems and networking [17] and drug design [18]. Our proposed Deep RL solution is a Multi-Agent model-free RL method, a configuration with multiple independent agents which allows the solution to be scaled to any telescope diameter, only limited by computational power, and without the use of any explicit model of the atmospheric dynamics.

Moreover, our proposed AO controller implements another deep neural network, an autoencoder, trained offline via Supervised Learning to mitigate the effect of noise in the WFS data, an approach we previously described in [19], so that the RL agents can focus on reducing other types of errors. Autoencoders [20] are neural networks that compress an input (e.g. an image) into a smaller size latent representation and reconstruct this representation to its original size . A possible use-case for autoencoders is denoising which has been shown in applications such as denoising speech signals [21] or medical images [22].

The use of RL to enhance the performance of AO has been already studied. The work of [23,24] used RL controllers but was limited to WFS-less systems. More recently, in parallel to our research, two studies have emerged. [25] proposes an AO controller with a single agent model-based RL for a classical AO setting, a 23x23 SH-WFS installed on an 8m telescope. The model-based RL method reported in [25] is known for being able to learn faster than model-free approaches, however, model-free methods usually achieve better final performance [26] and have a lower inference time (as reported in [27] Chapter 8.8) which is crucial in the AO setting. Moreover, without our proposed Multi-Agent configuration, it is not clear that their approach could scale to a larger number of subapertures due to the so-called *curse of dimensionality* where the number of possible outputs grow exponentially with the number of features in the state increasing the complexity of the problem. [28] presents a model-free RL method for high-contrast imaging. Their model outperforms an integrator controller for closed-loop control both in a system with just a tip-tilt mirror or in an 8m telescope equipped with a 41x41 high-order DM. The differences between our works are the following: (1) we overview the challenges a RL controller must overcome in AO, (2) their work operates on a zonal actuator basis. To avoid the *curse of dimensionality*, they use CNN to leverage spatial patterns and a matrix of rewards, one per actuator. Instead, we use Multi-Agent systems where each agent independently controls a subset of global modes. (3) their choice of RL method is Deep Deterministic Policy Gradient (DDPG) [29] while we use Soft Actor Critic (SAC) [30], which is known to outperform DDPG. The main difference between both methods is that SAC maximises not only the cumulative reward but the entropy of the learnt controller, which usually makes it less sensitive to hyperparameter selection and more robust in general, (4) we provide a full error budget analysis and show that our algorithm reduces temporal error and, to a lesser extent, the propagation of aliasing, (5) on contrast to their work, we show that our proposed system is robust to noise and (6) we provide an analysis of the inference times demonstrating a credible path towards a real-time implementation.

Overall, the contributions of this paper are the following: (1) proposal for Multi-Agent model-free RL for AO control able to scale to a realistic extreme AO (XAO) setting, with 40x40 SH-WFS, on a *state-of-the-art* 8m diameter telescope, (2) insight on the different key problems that one must address to develop a successful RL Controller for AO, (3) analysis of the robustness of the RL method under different atmospheric conditions and dynamically evolving conditions, (4) analysis of the real-time capabilities of the RL approach, demonstrating its applicability to existing AO systems, (5) robustness under high levels of noise due to the autoencoder and (6) comparison against a *state-of-the-art* LQG method with perfect knowledge of atmospheric conditions and evidence that our data-driven approach achieves similar performance.

The remaining parts of the paper are organised as follows: in Section 2 we recall the basics of a typical closed-loop AO setting with a classical integrator controller and we present the modal basis used for our work. In Section 3.1 we recall briefly the theoretical background supporting RL and propose a straightforward implementation in AO. In Section 3.2 we explain why this straightforward implementation fails and the challenges a RL approach must address to be successful for this application. In Section 3.3 we detail the final implementation of our approach. Finally, in sections 4 and 5., we provide the results obtained with realistic numerical simulations, derive conclusions and discuss future work.

2. Background

2.1. Classical controller with an integrator

In classical AO, the baseline control approach is realised by combining a linear wavefront reconstructor and an *integrator law*. At each iteration of the closed-loop, t, a measurement vector $\boldsymbol{m}_t$ (note that vectors are denoted with bold letters across the article), is computed from the SH-WFS e.g. via the pixel information from each subaperture and the computation of centroids for each spot image. Then, a simple linear relation gives the so-called residual commands $\boldsymbol{c}_t$ to be applied on the DM actuators to solve the wavefront reconstruction problem:

$$\boldsymbol{c}_t = R \cdot \boldsymbol{m}_t, \tag{1}$$

where R is usually referred to as the command matrix. There are many approaches to compute R, the simplest one being the Least Square inverse of the so-called interaction matrix (i.e. the response matrix of the wavefront sensor to each degree of freedom of the DM). The integration of residual commands with previous commands applied on the DM, $\boldsymbol{C}_{t-1}$, using a weight, the so-called gain g, mitigates the effects of this imperfect wavefront reconstruction and gives the final expression for a command at timestep t, C_t, for the integrator controller as seen in Eq. (2).

$$\boldsymbol{C}_t = \boldsymbol{C}_{t-1} - g \cdot \boldsymbol{c}_t, \tag{2}$$

where the gain g, has to be optimised to maximise the AO system performance.

2.2. B_{tt} modal basis

The concept of modal basis inherits from the work of [31], in which the use of Zernike polynomials was introduced to decompose complex optical aberrations into a set of orthonormal (or quasi-orthonormal) modes and the statistical properties of the temporal evolution of these modes for aberrations induced by atmospheric turbulence were exposed.

In this work, we consider the modal basis described in [32], and called hereafter B_{tt} basis. The choice of this particular modal basis stems from the following properties: (1) it spans the full DM actuator space, (2) all modes are orthogonal (whereas in the actuator space due to coupling between actuators they are not), (3) piston and tip-tilt are filtered out, which allows tip-tilt being only controlled by a specific tip-tilt mirror, and (4) the basis is normalised over the pupil surface. The B_{tt} basis derivation is recalled in Section 1 of Supplement 1. In this paper we noted the representation of commands in the B_{tt} space as $\hat{\boldsymbol{C}}$ ($\hat{\boldsymbol{c}}$ for the residual commands), where $\boldsymbol{C} = B_{tt} \cdot \hat{\boldsymbol{C}}$. The B_{tt} basis provides a space for the RL controller to operate on with the ability to filter out some of these modes for which the SH-WFS has low to no sensitivity (e.g. the so-called waffle modes). Additionally, an accurate error breakdown can be obtained in the same space.

3. Adaptive optics control with reinforcement learning

3.1. Reinforcement learning and a direct implementation in adaptive optics

Reinforcement Learning [27] addresses the problem of agent-environment interaction. At each timestep, t, the agent receives an observation state vector from the environment, $\boldsymbol{s}_t = \boldsymbol{s}$, executes an action vector, $\boldsymbol{a}_t = \boldsymbol{a}$, and the environment changes resulting in a new observation state vector, $\boldsymbol{s}_{t+1} = \boldsymbol{s}'$, and a reward, $r_{t+1} = r$, a scalar that measures agent's performance. The RL setting is formalised as a Markov Decision Process (MDP) with the following elements: (1) a set of states, $\mathbb{S}$ ($\boldsymbol{s} \in \mathbb{S}$), (2) a set of actions $\mathbb{A}$ ($\boldsymbol{a} \in \mathbb{A}$) (3) a reward function $R(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ and (4) a probability transition function $P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ between states upon taking an action.

The goal of RL is to learn, via trial and error, a function named policy, $\pi(s)$, that maps states to actions and which maximises the cumulative sum of the reward, called return, J. The objective of finding the optimal policy, π^*, can be expressed as finding the policy that maximises J in expectation:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_\pi \left[J\right] = \operatorname*{argmax}_{\pi} \mathbb{E}_\pi \left[\sum_{t=0}^{t=T-1} \gamma^t r_{t+1}\right], \tag{3}$$

where γ is a discount factor that weighs future rewards and T is the timestep where the task ends.

There are two main families of RL methods: model-based RL, which use a function that expresses the dynamics of the environment usually learned via interactions with the same environment, and model-free RL which do not. It is important to distinguish traditional model-based predictive controllers, where the model is constructed by systems of equations and prior assumptions, and model-based RL where the model is learned purely from data. In the following sections, we will argument why the choice of model-free RL might be better suited for AO control. The *state-of-the-art* model-free RL is Soft Actor Critic (SAC) [30] that augments the RL objective with an entropy term which allows the method to become more robust to hyperparameter selection. We have not contributed to the general theory of SAC, but for readers with no background in RL, a description of this approach is included in Supplement 1 Section 2.

For any RL problem, the tuple $(\boldsymbol{s}, \boldsymbol{a}, r)$ must be defined to correctly describe the environment. A straightforward definition of those terms for a closed-loop AO system equipped with a SH-WFS could be one that is similar to the integrator approach. For a timestep t, (1) the state, $\boldsymbol{s}_t$, corresponds to the measurements computed from the WFS, $\boldsymbol{s}_t = \boldsymbol{m}_t$. (2) the action, $\boldsymbol{a}_t$, corresponds to the commands to be applied to the DM actuators, $\boldsymbol{a}_t = \boldsymbol{C}_t$. (3) the reward, r_{t+1}, corresponds to a function to drive the RL controller to minimise WFS measurements $r_{t+1} = -||\boldsymbol{m}_{t+1}||_2^2$ where $|| \cdot ||_2$ indicates the L2 norm. Note that, as RL maximises rewards, we have to include a minus sign in the reward proposal.

Figure 1 shows a graphical description of an implementation of such system. However, this definition does not take into account many of the challenges of applying RL to AO and, from our experience, does not perform well outside of effortless configurations for the telescope and atmospheric conditions. In the next sections, we provide a careful analysis of the different challenges related to developing a RL controller for *state-of-the-art* AO systems and use it to derive a pragmatic approach more relevant to this kind of settings.

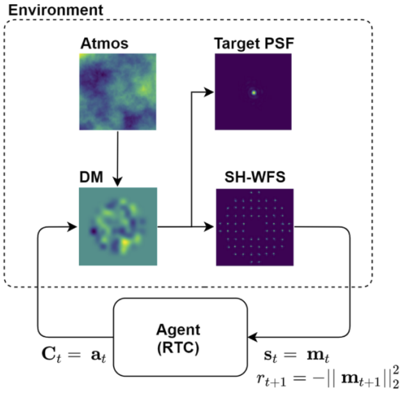

Environment
Atmos
Target PSF
DM
SH-WFS
Agent (RTC)
$\mathbf{C}_t = \mathbf{a}_t$
$\mathbf{s}_t = \mathbf{m}_t$
$r_{t+1} = -|| \mathbf{m}_{t+1}||_2^2$

Fig. 1. Agent-environment interaction with a direct RL approach. The environment corresponds to the atmosphere, the DM and the WFS while the agent corresponds to the RTC.

3.2. *Challenges of applying RL to AO*

In this section, we provide an overview of the challenges of applying RL to practical applications, as identified in [33], and analyse them from the AO perspective.

3.2.1. Real-time inference

Considering the typical coherence time of atmospheric turbulence on the best astronomical sites, an AO controller is expected to have an end-to-end latency better than about 2 *ms*. In AO controllers based on RL, the inference time corresponds to the time taken to compute the corresponding action, a, and it may vary depending on the RL method used. For instance, the model-based RL approach described in [25] starts by learning a model of the dynamics of the environment. Then, to compute an action, it samples candidate actions from a given distribution, propagates them through the learned model and selects the one that leads to maximum cumulative reward. Due to this repeated propagation and sampling, such approaches have a large inference time (up to 120 *ms* on consumer hardware and for an AO system with 448 actuators as reported on [25]) most certainly not compatible with the basic AO requirement mentioned above.

This is one of the main drivers to use **model-free** RL since such methods only need to do a single forward pass over their policies to select an action. From the different model-free methods we have elected **SAC** because of its robustness and performance.

3.2.2. Training time

In general, a RL agent needs to process a high number of tuples $(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s}')$ to learn a useful control policy. During the learning process, the agent produces exploratory actions that can lead to a low AO performance. In order to avoid wasting on-sky time, we need to minimise the number of samples required for learning. To do so, we consider the **residual policy learning** approach [34,35] which learns a correction on top of an existing controller that is not perfect. In the context of the AO application, our RL method will learn a correction on top of the integrator controller. On timestep t, the modification by the RL agent with an action, $\boldsymbol{a}_t$, on the *integrator law* from Eq. (2) is:

$$\boldsymbol{C}_t = \boldsymbol{C}_{t-1} - g \cdot \boldsymbol{c}_t + \boldsymbol{a}_t. \tag{4}$$

3.2.3. High dimensional continuous state-action spaces

A recurrent problem of many ML methods is the *curse of dimensionality* [27] where the number of possible combination of state-action, hence the problem complexity, grows exponentially with the increase in the dimensions of the problem.

While high dimensional state spaces is a common occurrence in RL problems, high dimensional action spaces is not, as most benchmarks have few integer or continuous actions. Nevertheless, some studies have focused on high-dimensional action spaces such as in [36] where each action dimension is treated with some degree of independence as if each action dimension was an independent agent and worked with the others to achieve maximum reward in what is known as **Multi-Agent RL (MARL)**. Following this work, we propose to reformulate the state, action and reward representation suggested above to match the AO setting with a MARL problem, after providing a brief formal review of MARL.

MARL is an extension of RL with N agents interacting with the environment simultaneously. In this work, we consider a special case of MARL: decentralised cooperative MARL (Dec-MDP) [37] where each agent cooperates to maximise a global reward without a central entity that guides all the agents. The work of [38] gives an analysis of the different components of a Dec-MDP and serves as our starting point. The main initial difference with a MDP is that $\mathbb{A}$ expresses a set of joint actions, $\boldsymbol{a}$, with each agent, i, having its local action, $\boldsymbol{a}^i$, (and the corresponding local policy, π^i) independent of other agents. A joint action can be expressed as $\boldsymbol{a} = (\boldsymbol{a}^1, \boldsymbol{a}^2, \ldots, \boldsymbol{a}^N)$.

Depending on the peculiarities of the problem addressed, a set of properties can be derived. Given a timestep, t, a Dec-MDP is said to be **factored** if the states can be separated in N components (one per agent) where each agent, i, has its local state, $\boldsymbol{s}^i_t$, independent of other agents and the joint state, $\boldsymbol{s}_t$, is formed by concatenating local states $\boldsymbol{s}_t = (\boldsymbol{s}^1_t, \ldots, \boldsymbol{s}^N_t)$. A Dec-MDP is said to be **transition independent** if the transition probability function of local state i, $\boldsymbol{s}^i_t$, only depends on previous value of local state i, $\boldsymbol{s}^i_t$, and local action i, $\boldsymbol{a}^i_t$. A Dec-MDP is said to be **reward independent** if the reward function can be factored in a set of N independent components. Again, each agent will have its local reward, r^i_{t+1}, independent of other agents. If the Dec-MDP is factored, transition independent and reward independent we end up with **N independent MDP problems**.

This will be the basis to turn the AO problem from a complex high-dimensional one into N simpler lower-dimensional ones. If we use the residual commands in the B_{tt} space, $\hat{\boldsymbol{c}}$, instead of the WFS measurement space, and since the modes are orthogonal, the state will already be factored and transition independent. Moreover, we can choose a reward that is a function of the residual commands $r = f(\hat{\boldsymbol{c}})$ providing, thanks to modes orthogonality, the reward independence we are looking for. Having a factored, transition independent and reward independent problem will allow us to have the desired N independent MDP problems which will circumvent the *curse of dimensionality*.

3.2.4. Delay and characterising the atmospheric conditions

The loop delay is another characteristics that should properly be taken into account, beyond the direct approach proposed above. In an AO system, there is usually a delay for the corrections to be observed on the target PSF, and to handle that, the agent needs to have some information regarding the fact that commands computed at a given step will be realised by the DM in the future. Besides, a single frame may not be enough to correctly characterise the atmospheric turbulence dynamics with the RL controller. For instance, it is not possible to estimate the direction of evolution of an equivalent phase screen with a single frame. Therefore, to correctly depict the delay and the dynamics of the atmosphere the agent will include an **history of commands and/or measurements**.

In addition to tackling the impact delay for the state representation, we can also leverage delay information for the update. As explained in Supplement 1 Section 2, SAC uses the concept of experience replay in which a memory of tuples $(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s}')$ is expanded as more experience is obtained and is used to update the agent. If the delay in the system is known, the reward, r, and next state, $\boldsymbol{s}'$, can be obtained by **waiting for a number of timesteps corresponding to this delay**.

3.2.5. Noise

Another challenge comes from read-out and photon noise. Since the proposed controller utilises measurements and/or residual integrator commands which may be corrupted by noise or its propagation, noise could become a severe impediment to proper training of the RL controller. In order to mitigate this effect, we leverage the approach, already published in [19], of denoising wavefront sensor images with deep neural networks.

An autoencoder is an architecture that compresses an input image into a latent representation and decompress it to its original size. Figure 2 depicts the **autoencoder** architecture used for denoising a WFS image. The autoencoder is trained via Supervised Learning with a dataset consisting of pairs of images (noisy, noise-free).

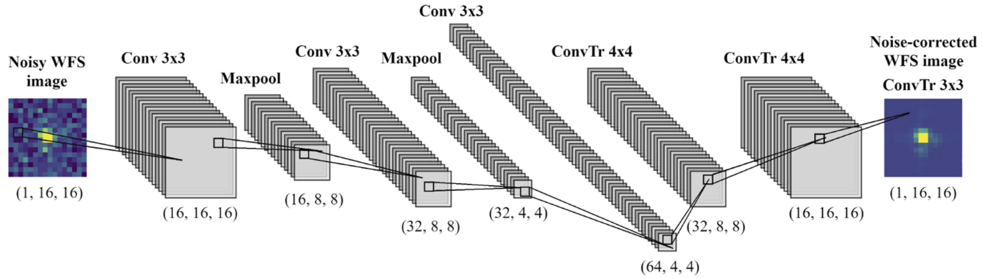


Fig. 2. Autoencoder for denoising WFS images. The tuple indicates the shape of the output at each step.

While, as opposed to the RL agents, the autoencoder will be trained offline by a Supervised Learning procedure, the main motivation to use it comes from the fact that in [19] it was shown that the denoising did not depend on the current atmospheric conditions but only on the noise levels. For that reason, just by having a set of pre-trained networks and choosing the correct one for the current noise level would solve the issue. By doing so, the RL controller can be fed with noise-free data, as much as possible, and can thus be used to tackle all the other sources of error.

3.2.6. Dynamically evolving atmospheric conditions

Finally, while most simulations are conducted with a set of fixed atmospheric parameters, a more realistic experiment would consider atmospheric parameters in constant evolution. In particular, the Fried parameter, wind velocity, and wind direction may evolve and affect the controller during the course of an observing run. Fortunately, as our RL controller learns online from telemetry data, it can adapt to atmospheric changes without any further modification. To test the robustness of adaptation, **we have conducted a number of experiments emulating dynamically evolving atmospheric conditions and will present the results later in the paper.**.

3.3. Final implementation of MARL for AO

Gathering the challenges and solutions we proposed in Section 3.2 the final implementation can be described as follows:

- The problem is divided into N subproblems where for each subproblem there exist an agent, i, that controls a subset $\boldsymbol{M^i}$ of B_{tt} modes.

- For timestep or frame t, the state for agent i, $\boldsymbol{s}_t^i$, consists of the current residual command in the B_{tt} space for the modes controlled by this given agent, $\hat{\boldsymbol{c}}_t^i$, and the history of commands in the B_{tt} space for the same modes.

$$\boldsymbol{s}_t^i = (\hat{\boldsymbol{c}}_t^i, \hat{\boldsymbol{C}}_{t-1}^i, \hat{\boldsymbol{C}}_{t-2}^i, \ldots, \hat{\boldsymbol{C}}_{t-n}^i). \tag{5}$$

  In addition, we propose a second approach for the state definition which differs slightly from the one above. We found out that better AO performance can be achieved if, for a given agent, the state is built not only from the modes it controls but using neighbouring modes as well. We call this approach "windowed" MARL as it uses a window of size N_w to "look" at neighbouring modes. In our experiments, after some fine tuning through trial and error, we chose to use $n = 3$ and $N_w = 20$. These values have been obtained through a rather empirical process, and probably require a more formal analysis to be connected to system complexity and turbulence parameters.

- For timestep or frame t, a joint action $\boldsymbol{a}_t$, formed by the combination of all the agents actions, $\boldsymbol{a}_t^i$, as $\boldsymbol{a}_t = (\boldsymbol{a}_t^1, \ldots, \boldsymbol{a}_t^N)$, is injected into the closed-loop via a correction term (as described in Eq. (4)) on the integrator controller in the B_{tt} space:

$$\boldsymbol{C}_t = B_{tt} \cdot (\hat{\boldsymbol{C}}_{t-1} - g \cdot \hat{\boldsymbol{c}}_t + \boldsymbol{a}_t). \tag{6}$$

- Model-free *state-of-the-art* SAC is chosen as the RL method. In SAC, a policy parameterised with a neural network is trained to obtain the parameters of a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ that maximises the SAC objective (maximising cumulative reward and policy entropy). During the exploration phase, actions are sampled from the Gaussian distribution to find the state-action pairs that lead to the best cumulative reward. Once trained, we evaluate the actions without any exploration at all, which will simply be the mean of the Gaussian distribution.

Moreover, in SAC each pair $(\boldsymbol{s}, \boldsymbol{a})$ has to be assigned to a pair $(r, \boldsymbol{s'})$. Assuming a simulation in which there is d-1 frame(s) of delay (where frame here means minimum integration time on the WFS) from computing the commands to actually changing the shape of the DM and d frame(s) of delay from computing the commands to observing its effect on the WFS image and PSF we need to wait d frames to assign $(\boldsymbol{s'}, r)$ to $(\boldsymbol{s}, \boldsymbol{a})$. At timestep t, for state $\boldsymbol{s}_t$ and action $\boldsymbol{a}_t$, the correct indices for the next state and reward will be $\boldsymbol{s}_{t+d}$ and r_{t+d}. This is sketched in Fig. 3 for an example with 2 frames of delay.

Finally, SAC uses the concept of experience replay. At each iteration, a tuple $(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s'})$ is saved into a memory, $\mathcal{D}$. Then, at update time, this memory is used to sample mini-batches of experiences to update π. In the MARL case, each agent, i, has its own memory $\mathcal{D}^i$ and obtains data under the form of tuples $(\boldsymbol{s}^i, \boldsymbol{a}^i, \boldsymbol{s'}^{,i}, r^i)$.

- For timestep or frame t, the reward function for agent i, r^i, reflects the goal of minimising the residual commands in the B_{tt} space, $\hat{\boldsymbol{c}}$, for the modes controlled by this agent, $\boldsymbol{M^i}$, and is built as:

$$r_t^i = -\frac{\kappa}{|\boldsymbol{M^i}|} \sum_{m \in \boldsymbol{M^i}} (\hat{\boldsymbol{c}}_t^m)^2, \tag{7}$$

  where κ is a constant to improve numerical stability of the algorithm. The value of κ depends on the units of $\hat{\boldsymbol{c}}$, from empirical considerations and the units system in our simulation tool, we have chosen to set it to 1000.

In addition to the main points above, we need to take into account preprocessing. Feature scaling is a must in ML methods and for that reason, we record command and measurement data from 20000 frames using the integrator on the same AO system and scale it with z-score normalisation. Moreover, we set a range on the maximum correction the actions can bring which will be at most ±5% of the peak to valley of the integrator command values for those 20000 frames. Finally, we do an additional test to discard modes for which the WFS has low sensitivity (e.g. waffle modes). This is done empirically by maximising the Long Exposure (L.E.) Strehl Ratio (SR) over 1000 frames while discarding a variable amount of modes.

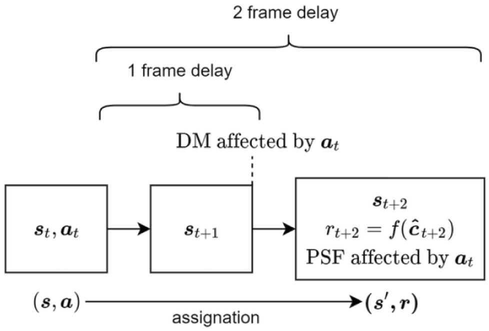

2 frame delay
1 frame delay
DM affected by $\boldsymbol{a}_t$
$\boldsymbol{s}_t, \boldsymbol{a}_t$
$\boldsymbol{s}_{t+1}$
$\boldsymbol{s}_{t+2}$
$r_{t+2} = f(\hat{\boldsymbol{c}}_{t+2})$
PSF affected by $\boldsymbol{a}_t$
$(\boldsymbol{s}, \boldsymbol{a})$
assignation
$(\boldsymbol{s}', \boldsymbol{r})$

Fig. 3. Temporal notation for current state and action, and next state and reward for the example of 2 frames of delay, 1 frame delay for the DM and an additional frame delay for the WFS image and PSF computation. Each block indicates a step in the AO loop. At the beginning of each step, a raytracing operation through the atmosphere and the AO system is producing a phase screen for both the WFS and target PSF and the measurement and command vectors are computed. At the end of each step the DM shape changes based on the commands computed d-1 steps before. The WFS image (which the commands are computed from) and PSF are computed from the results from the initial raytracing operation which considers the DM shape in the previous step, therefore, both operations have an additional frame of delay.

In the Multi-Agent method explained above, training and inference (of action selection) are performed simultaneously (as usually in RL) but in different computational processes. This parallelism allows the proposed controller to work in real-time. A sketch of the agents-environment interaction is shown in Fig. 4 and the pseudocode for the training process is provided in Supplement 1 Section 3.

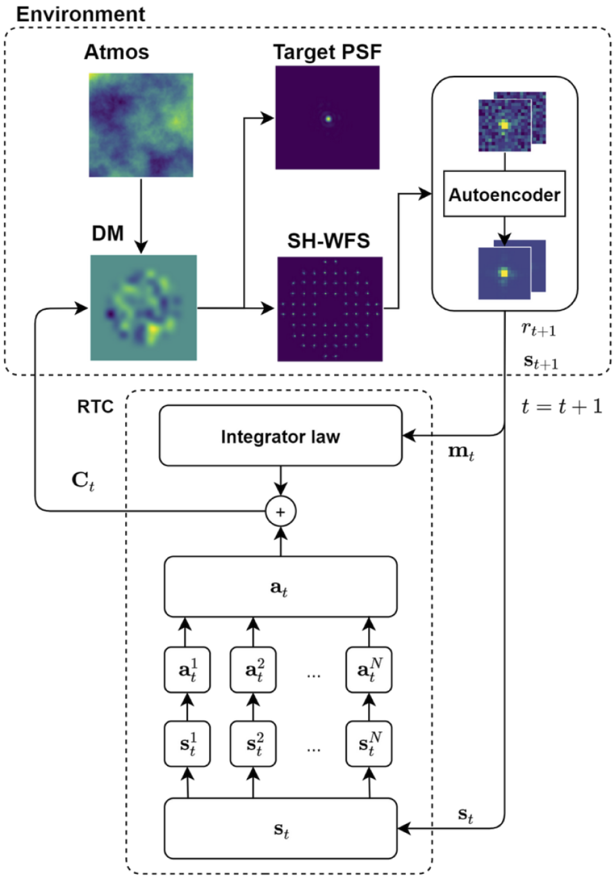


Fig. 4. Multi-Agent Reinforcement Learning for Adaptive Optics control. The environment, as seen from the Multi-Agent RL controller, is composed of the atmosphere, the DM, the WFS and the autoencoder that denoises the WFS image. The Multi-Agent RL will take the state as an input, separate it in N components and produce an action for each component. All these actions will be combined into a joint action and summed to the integrator controller output.

3.4. Error budget

When designing an AO system, one usually derives an error budget, which encompasses all the possible sources of errors and is expressed as a sum of independent residual phase variance terms, providing a framework to estimate which errors can be addressed using the various components of the AO system. Tools such as ROKET (erROr breaKdown Estimation Tool) [32], built on top of a high-performance end-to-end GPU-based simulator, COMPASS (COMputing Platform for Adaptive opticS System) [39], can provide an error breakdown for a given AO system configuration. However, so far, this tool has been designed and implemented considering an integrator controller with a single frame of delay. In this section, we will derive a ROKET extension that works for a non-linear correction on top of the integrator controller with d frame(s) of delay.

We consider a system with a single SH-WFS that uses a natural guide star and where the integrator controller is corrected by a generic RL method. Consider the residual wavefront phase observed by the WFS, Φ, which is the sum of turbulent, Φ_{turb}, and DM, Φ_{DM}, phases, expressed as a sum of B_{tt} modes:

$$\Phi_{turb} = \sum_i b_i B_{tt,i}, \qquad \Phi_{DM} = \sum_i v_i B_{tt,i}. \tag{8}$$

Given the interaction matrix, D, d frame(s) of delay and the vector of B_{tt} coefficients for the following contributors: the turbulent phase, $\boldsymbol{b}$, the DM phase, $\boldsymbol{v}$, propagated aliasing, $\boldsymbol{l}$, as well as noise, $\boldsymbol{n}$, and other non-linear terms, $\boldsymbol{u}$, the WFS measurement vector, $\boldsymbol{m}$, for frame k can be written as:

$$\boldsymbol{m}_k = D\boldsymbol{b}_k + D\boldsymbol{v}_{k-d} + \boldsymbol{l}_k + \boldsymbol{n}_k + \boldsymbol{u}_k, \tag{9}$$

where $\boldsymbol{v}$ can be obtained from the linear integrator solution corrected (as in Eq. (4)) from the RL actions vector, $\boldsymbol{a}$:

$$\boldsymbol{v}_k = \boldsymbol{v}_{k-1} - gR\boldsymbol{m}_k + \boldsymbol{a}_k. \tag{10}$$

The error vector at each frame, $\boldsymbol{\epsilon}_k$, is the sum of turbulent and DM phases considering a delay d:

$$\boldsymbol{\epsilon}_k = \boldsymbol{b}_k + \boldsymbol{v}_{k-d}. \tag{11}$$

If we expand Eqs. (9), (10) and (11) we obtain:

$$\boldsymbol{\epsilon}_k = \boldsymbol{\epsilon}_{k-1} - gRD\boldsymbol{\epsilon}_{k-d} + (\boldsymbol{b}_k - \boldsymbol{b}_{k-1}) - gR(\boldsymbol{l}_{k-d}) - gR(\boldsymbol{n}_{k-d}) - gR(\boldsymbol{u}_{k-d}) + \boldsymbol{a}_{k-d}. \tag{12}$$

The difference, other than the delay-corrected terms, between this expression and the one in [32] is the RL term (a_{k-d}). From (12), we can isolate five different error breakdown contributors:

$$\boldsymbol{\epsilon}_k = \boldsymbol{\beta}_k + \boldsymbol{\rho}_k + \boldsymbol{\eta}_k + \boldsymbol{\mu}_k + \boldsymbol{\zeta}_k, \tag{13}$$

where $\boldsymbol{\beta}$ is the **bandwidth error** that appears as a result of the global loop delay, $\boldsymbol{\rho}$ is the **aliasing error** which appears due to perturbations at frequencies higher than the Nyquist limit defined by the WFS spatial sampling, $\boldsymbol{\eta}$ is the **noise error** which is related to read-out noise from the detector and from the photon statistics, $\boldsymbol{\mu}$ is the **wavefront deviation error** which is composed of all other typical sources of error in an AO system, such as non-linearities in WFS measurements due to diffraction errors on the SH-WFS, and $\boldsymbol{\zeta}$ is the **RL term** which represents the RL controller contribution to the error budget. While the RL term is not an error strictly speaking, it can be used to quantify how the RL controller modifies the error budget and interacts with other sources of error. The expression of the contribution for each term is given by the Eqs. (14) to (18).

$$\boldsymbol{\beta}_k = \boldsymbol{\beta}_{k-1} - gRD\boldsymbol{\beta}_{k-d} + (\boldsymbol{b}_k - \boldsymbol{b}_{k-1}). \tag{14}$$

$$\boldsymbol{\rho}_k = \boldsymbol{\rho}_{k-1} - gRD\boldsymbol{\rho}_{k-d} - gR\boldsymbol{l}_{k-d}. \tag{15}$$

$$\boldsymbol{\eta}_k = \boldsymbol{\eta}_{k-1} - gRD\boldsymbol{\eta}_{k-d} - gR\boldsymbol{n}_{k-d}. \tag{16}$$

$$\boldsymbol{\mu}_k = \boldsymbol{\mu}_{k-1} - gRD\boldsymbol{\mu}_{k-d} - gR\boldsymbol{u}_{k-d}. \tag{17}$$

$$\boldsymbol{\zeta}_k = \boldsymbol{\zeta}_{k-1} - gRD\boldsymbol{\zeta}_{k-d} + \boldsymbol{a}_{k-d}. \tag{18}$$

To validate the extension of ROKET we we will reconstruct the PSF from the given terms and compare with the one given by the simulator as the sum of all the instantaneous (or Short Exposure, SE) PSF obtained at each iteration.

4. Results

4.1. Simulation parameters

In this section, we enumerate the different parameters for the AO simulation and the hyperparameters for the MARL controller. We used a combination of COMPASS [39] for simulating the AO system and ROKET [32] for error analysis. Table 1 shows the parameters for the system architecture, the atmosphere and the target.

Table 1. Constant AO parameters among the different simulations.

Telescope parameters		**Atmospheric parameters**	
Diameter (m)	8	Num. layers	3
AO loop parameters		Altitude per layer (km)	0/4.5/14
Loop frequency (Hz)	500	r_0 (m)	0.16 @ 500 nm
Frames of delay	2	r_0 distribution	0.6/0.25/0.15
Target parameters		L_0 (m)	10^5
λ_{target} (μm)	1.65	**WFS parameters**	
DM parameters		Num. subapertures	40x40
Mirrors	PZT and TT	Pixels per subaperture	16
Coupling (PZT)	0.2	Pixel size	0.25
Num. Corrected B_{tt} modes	1262	λ_{wfs} (μm)	0.5
Num. Total B_{tt} modes	1283	Read-out noise	0 or 3 e- RMS

Our experiments are targeting XAO systems on 8m telescopes equipped with a 40x40 SH-WFS. The SH-WFS uses a simple Center of Gravity (CoG) algorithm for the WFS measurements. The gain for the integrator controller, as well as the number of low sensitivity modes to discard, have been optimised manually before every experiment. For simplicity purposes, noise will only be active if we use the autoencoder to test the robustness of our system against this source of errors.

The atmospheric turbulence is composed of 3 equivalent layers in our simulation which provides a good trade-off between complexity and ease of use. As we want to test robustness under different atmospheric conditions we have proposed several configurations both for wind speed and direction of each of the layers. The different values can be found in Table 2.

Table 2. Different configurations of wind speed (v) and wind direction for each layer.

	Dir. 1 (°)	Dir. 2 (°)	Dir. 3 (°)	v_1 (m/s)	v_2 (m/s)	v_3 (m/s)
Layer 1	0	0	0	20	15	10
Layer 2	0	15	45	15	10	5
Layer 3	0	30	90	25	20	15

Finally, RL experiments are usually defined in episodes of a certain length between which the simulation resets. We have decided to do the training experiments with episodes of 1000 frames. Every 50 episodes, an evaluation episode is conducted where we switch off the exploration of the RL agents, and the performance of RL and integrator controllers compared.

Regarding MARL hyperparameters, we used 43 agents per experiment (42 controlling 30 modes each and a single agent controlling the tip-tilt mirror). We chose bins of 30 modes per agent because, in our early experiments, we observed that controlling >50 modes per agent led to reduced performance (due to high state-action dimensionality). Moreover, we decided to separate the tip-tilt modes from the others and drive them with a dedicated agent in good alignment with the typical AO system architecture that has a dedicated tip-tilt mirror. In order to ease the implementation, we assigned to the piezoelectric DM the same number of modes to each agent. Out of the overall 1283 modes obtained from the B_{tt} decomposition corresponding to this system configuration, only 1262 were corrected by the MARL approach and the remaining ones were either controlled by the integrator (16 modes) or filtered out as low sensitivity modes (5 modes).

To verify that our MARL controller is robust among different initialisations, we ran all the learning experiments for 3 different initial seeds. Following the default implementation of SAC, all the neural networks architectures used are fully connected ones. Additional RL hyperparameters can be found in Supplement 1 Section 4. The autoencoder hyperparameters are the same as the ones used in [19].

4.2. Experiments

In this section, we aim to provide answers to the following questions regarding the proposed controller:

1. Does the MARL controller outperform the integrator controller?
2. Is the MARL controller robust to different initial atmospheric conditions?
3. Is the MARL controller robust to dynamically changing atmospheric conditions?
4. How does the MARL controller cope with errors from the integrator controller, in particular regarding the gain value?
5. How does the autoencoder and MARL combination behave when noise is activated, and how robust is the autoencoder to different noise levels?
6. Which terms from the error budget are affected by the MARL controller? Are there any properties from the learned behaviour that we can observe?
7. How does the MARL controller compare against current *state-of-the-art* methods such as LQG?
8. Is the MARL controller compatible with an actual real-time implementation?

4.2.1. Robustness to different initial atmospheric conditions

In this section, we address both questions 1 and 2. Figure 5 shows the training curves comparing the MARL controller, the integrator controller and the windowed MARL for the nine possible configurations of wind direction and speed of Table 2.

The MARL controller is able to outperform the integrator in approximately 500k frames for all atmospheric conditions which is equivalent to 17 minutes of training time. This number could be significantly reduced if we finetune the RL hyperparameters related to training time or pre-train the networks with a realistic simulator configured to reproduce similar atmospheric conditions.

On the performance side, regarding wind conditions, the MARL controller is able to provide almost constant performance irrespective of the direction or speed. This is a clear advantage over the linear integrator as, for instance, for wind direction profile 1 and any wind speed profile the L.E. SR obtained with the RL controller is between 0.9 and 0.92 after 2 million steps, while the performance with the integrator goes from 0.9 in the case of wind speed profile 3 (v_3) to 0.84 in the case of wind speed profile 1 (v_1). This leads to two conclusions: (1) the MARL controller is, most probably, mitigating bandwidth error as it is able to maintain similar performance when the bandwidth error increases and (2) the MARL controller shines best when the bandwidth error is high.

Concerning wind direction conditions, we noticed that the gains introduced by the windowed approach were only noticeable if the wind direction was similar between layers. Our hypothesis

Optics EXPRESS

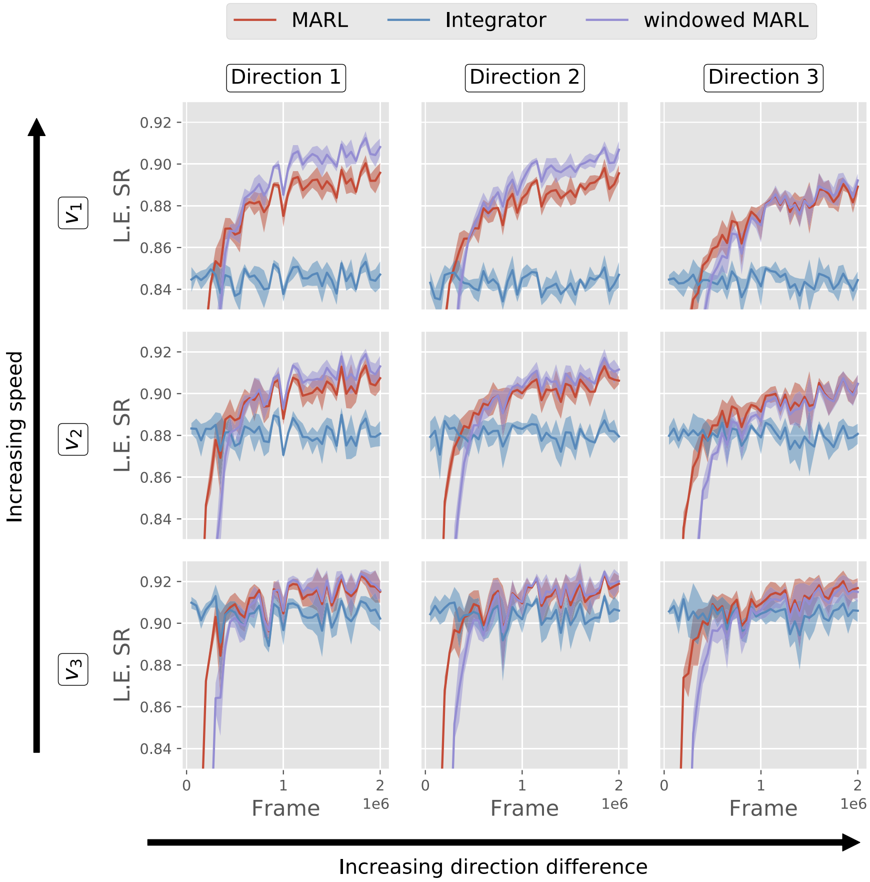


Fig. 5. Comparison of L.E. SR obtained in the evaluation episodes for MARL/windowed MARL and integrator controllers with different initial atmospheric conditions. Initially, the MARL controller has worse performance than the integrator controller, but after sufficient training, it surpasses the integrator controller in all cases.

stands as follows: when the wind direction is similar between layers, a stronger correlation appears between neighbouring modes and, consequently, the windowed approach can help the MARL controller to learn a more complex turbulence model since useful information can be shared between modes. In any case, as the only downside of the windowed approach seems to be a small effect on initial training time, it appears that it should be used at all times.

Overall the MARL controller appears robust to different initial atmospheric conditions with its performance being always better than the integrator.

4.2.2. Robustness to dynamically changing atmospheric conditions

In this section, we address question 3. We select a default parameter file with wind speed profile v_2 (15, 10, 20 m/s) and wind direction profile 1 (0, 0, 0 °) and we train a MARL controller for 1000 episodes and suddenly change the atmospheric conditions. The transitions in atmospheric conditions we consider are:

- Change in wind direction profile. Figure 6(a) shows the change in wind direction profile 1 to 2 and Fig. 6(b) shows the change in wind direction profile 1 to 3.

- Change in wind speed profile. Figure 6(c) shows the change in wind speed profile 3 to 2 and Fig. 6(d) shows the change in wind speed profile 2 to 3.
- Fig. 6(e) shows a change of the Fried parameter from r_0 = 0.16 m to r_0 = 0.08 m.

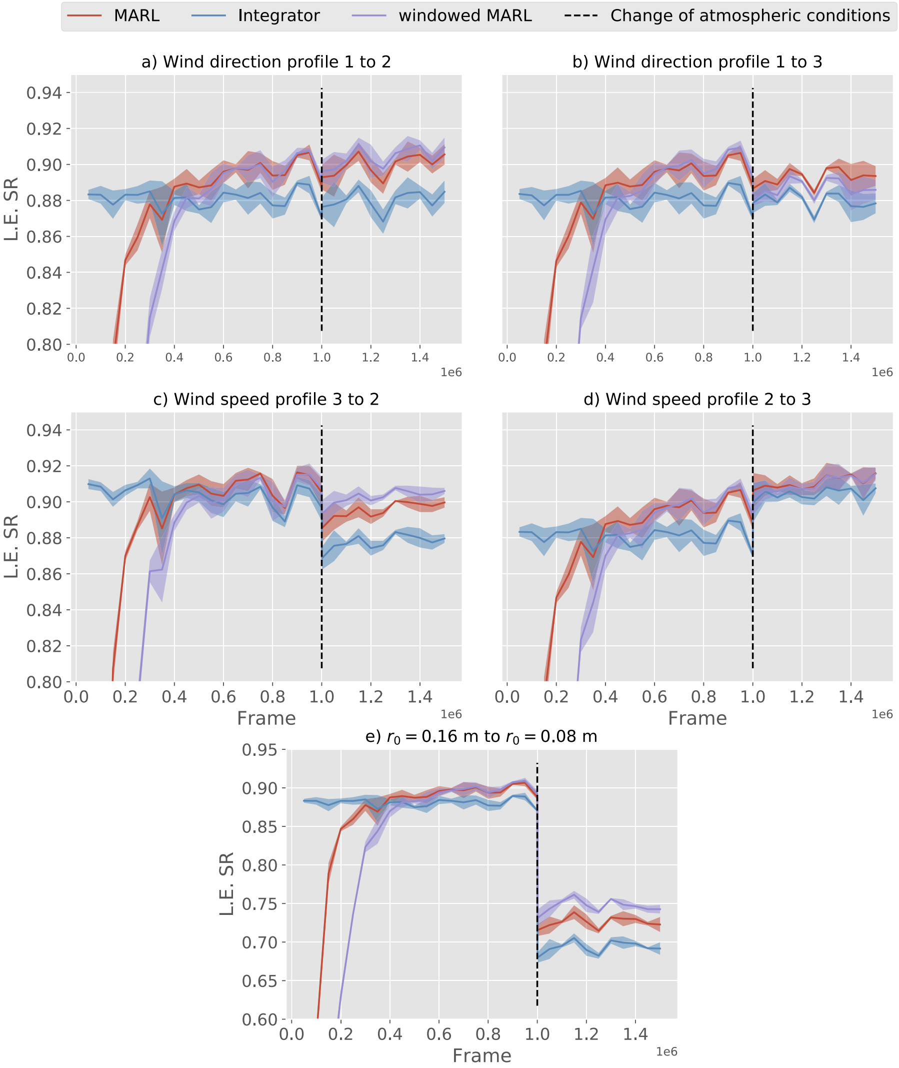


Fig. 6. Comparison of L.E. SR obtained in the evaluation episodes for MARL/windowed MARL and integrator controllers for experiments regarding changing atmospheric conditions.

As we can observe, the MARL controller outperforms the integrator in most experiments right after the transition from one condition to the other.

In the case of a significant change in wind direction profile (Fig. 6(b)), the MARL controller performance is significantly impacted with a stronger decrease for the windowed approach. Nevertheless, both MARL and windowed MARL controllers are able to adapt and achieve better SR (as compared to the integrator) as more data gets available. As we noted in the previous section, our hypothesis is that the windowed approach is able to exploit the correlation between modes in similar wind directions profiles but fails to do so in contrasting ones. Once the wind direction profile changes, those correlations can not be utilised and the windowed MARL approach will need time to adjust to have, at least, similar performance to the non-windowed one. This is however an extreme scenario and we do not expect a change in the wind direction profile in such an abrupt way during on-sky operations. Therefore, Fig. 6(a) provide more realistic insights on actual operational settings.

Regarding wind speeds, which is a clear contributor to bandwidth error, when transitioning from slow to fast wind in all layers, the MARL controller allows the AO loop to maintain almost constant performance, while the performance decreases significantly with the integrator controller. When the wind speeds for all layers decreases, hence does the contribution from the bandwidth error, the gain brought by MARL is less significant. This is another clue towards the fact that MARL is able to mitigate efficiently bandwidth error, while being rather robust to the evolution of turbulence properties.

Regarding the last experiment, in which we simulate an abrupt transition between a median r_0 value to bad seeing conditions (again an extreme scenario), MARL is able to adapt and provide the same relative performance gain compared to the integrator. This was expected from the remarks above on bandwidth error and considering that a good approximation of the turbulence coherence time can be obtained from the ratio between r_0 and the average wind speed (see e.g. Roddier 1981 [40]), the bandwidth error being thus inversely proportional to r_0. Interestingly, the windowed approach seems to adapt better to bad seeing conditions providing a significant boost in absolute performance compared to the simple MARL approach.

4.2.3. Robustness to errors in the integrator controller

In this section, we address question 4. Since we have designed the MARL approach to provide a correction term on top of the integrator, one may think that having an unoptimised integrator may lead to worse performance. Figure 7 shows the robustness of the MARL controller to different values of the integrator gain for a simulation with wind direction profile 1 and wind speed profile 2 (v_2).

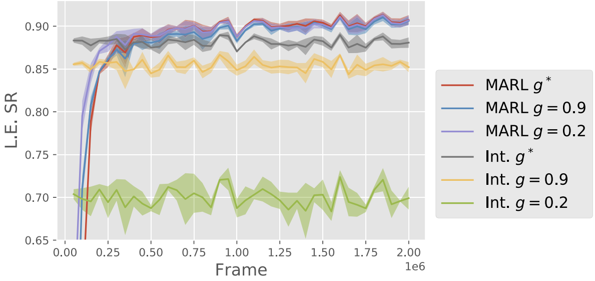


Fig. 7. Comparison of L.E. SR obtained in the evaluation episodes for MARL and integrator controllers for different values of integrator gain. The optimal gain is $g = 0.7$ and is denoted with an asterisk.

As shown in this figure, the MARL solution achieves approximately the same performance for very different gain values. This demonstrates an important property of MARL for on-sky operation since it shows that the performance boost provided by MARL does not depend on the optimisation of the integrator gain. As such, MARL could be seen as a way to guarantee the highest AO loop performance irrespective of actual turbulence conditions and corresponding integrator optimisations.

Similar properties have been evidenced in [25] where a model-based RL agent can adapt to miss-calibration of the interaction matrix.

4.2.4. Robustness to noise with the autoencoder

In this section, we assess the robustness to noise of the combination of MARL with an autoencoder addressing question 5. Again, for the sake of clarity, we assess the autoencoder performance given a single atmospheric configuration. In this case, we have tested the autoencoder in the most complex wind direction profile (3) and with wind speed profile 2 (v_2).

To generate data for training the autoencoder we built simulations with two WFS one with readout noise and photon noise and one without any noise at all. At each step of the simulation, the integrator is fed with the measurement vector from one of the WFS and images from both WFS are recorded. The integrator is getting data from the noisy WFS unless we are in high noise simulations (magnitudes > 10) since noise propagation in this case would be too high for the controller to produce relevant command vectors leading to a useless dataset at the output of the simulation. The full process to train the autoencoder is as follows:

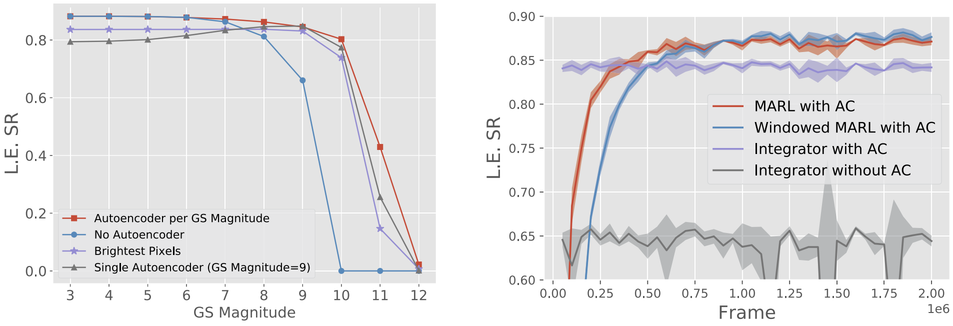
1. Choose the WFS that will be used to produce the dataset depending on the magnitude.
2. Optimise the gain of the integrator with no delay.
3. Record a sample of $1.2 \cdot 10^6$ image pairs (noisy, noise-free) for a given WFS subaperture in closed loop with the integrator and no delay.
4. Train the autoencoder with corresponding data with Supervised Learning methods.
5. Once trained, the learned autoencoder weights can be extrapolated to be used within a loop including e.g. 2 frames of delay. Note that, in that case, since we change the delay and include the autoencoder, the integrator gain must be reoptimised.

As a reminder from our previous paper [19], the autoencoder is used as a frontend to centroiding, to denoise images that are then processed using a simple Center of Gravity (CoG) approach to obtain the measurement vector from WFS images. We have tested the approach on a number of simulations, considering a mixture of photon and readout noise, with varying guide star magnitudes and constant readout contribution (3 e- RMS). We have trained one autoencoder per guide star magnitude.

In Fig. 8(a) we compare the performance for 1000 frames in terms of L.E. SR for an AO loop with 2 frames of delay and several control approaches: an integrator with an autoencoder trained for each guide star magnitude (in red), an integrator without autoencoder (in blue), and an integrator with a popular noise mitigation method, CoG on the brightest pixel (in purple). In the case of the latter, the number of brightest pixels has been optimised for each magnitude with a maximum of 11 brightest pixels. Moreover, in order to show the robustness of the autoencoder in the case where the magnitude it is trained on does not match the one of the guide source used during operations, we added to Fig. 8(a) the performance of an integrator on different guide star magnitudes but with an autoencoder trained only with data from magnitude 9 (in black).

The results indicate (a) that the autoencoder can provide a significant performance boost on the fainter end of our simulation campaign compared to the brightest pixel method and (b) that it is rather robust from the experiment involving training on a single guide star magnitude. The autoencoder is currently trained with data from all subapertures, including fully and partially illuminated ones, without discriminating them. As future work, we may train different autoencoders on different illumination levels to improve performance.

The next step is to combine denoising WFS images with an autoencoder, centroiding with a simple CoG approach and training the MARL to optimise the integrator performance. This would be the ideal setting for operations on sky, since it would not require any real-time fine tuning of parameters to optimise the AO loop performance. The results of such experiment are shown in Fig. 8(b) for a guide star magnitude of 9 and a readout noise of 3 e- RMS on the WFS,



(a) (b)

Fig. 8. a) Performance comparison for a 1000 frame AO loop with an integrator using various centroiding methods: simple CoG, brightest pixels CoG, autoencoder + simple CoG, where the autoencoder is trained with data from a single guide star magnitude, and autoencoder + simple CoG, where the autoencoder is trained for each guide star magnitude. b) Comparison of L.E. SR obtained in the evaluation episodes for MARL/windowed MARL and integrator controllers in the presence of noise and with an autoencoder for denoising purposes.

together with the results of a simple integrator working on noisy centroids (gain optimised) and an integrator fed by "denoised" centroids obtained with the same autoencoder (gain optimised as well). They show that both the simple and windowed MARL approaches are able to outperform the integrator even when the latter is fed with "denoised" data.

The results obtained with MARL but without autoencoder in this noisy setting are not reported since the loop performance in this case was not acceptable (i.e. below 0.1 L.E. SR). As the noise propagates to both state and reward information, the problem might be too complex for the RL agents to be able to learn a stable control policy.

4.2.5. Error budget analysis

In this section, we provide an error budget analysis for an AO system driven by a MARL controller after it is fully trained, using the ROKET tool and its extension described in Section 3.4. This analysis is based on a single set of simulation parameters (wind speed profile 2 (v_2), wind direction profile 1 with windowed MARL as a controller) for simplicity purposes. To use ROKET we record each component defined in Section 3.4 for 20000 iterations of a single simulation using a specific controller (integrator or MARL (without exploration)).

ROKET can reconstruct the PSF based on the combination of contributions from each error term described in Section 3.4, computed from data collected from the simulation. To verify that the proposed ROKET extension for the MARL controller is valid we compare the PSF produced by both ROKET and COMPASS in Fig. 9. As observed in this figure, these PSF profiles are almost identical, which confirms the validity of our ROKET extension.

The error breakdown provided by ROKET includes the covariances for all terms as well as cross-covariances between them. As seen in Fig. 10(a), in this simulation, the main contributors to the error budget (diagonal values) are: aliasing, bandwidth and RL terms. We recall the reader that the RL term is not an error per say but rather an expression helping in the analysis of which terms the MARL controller is affecting and correlated with. The covariance matrix of Fig. 10(a) is mostly diagonal, which means that for this given set of simulation parameters most of the terms are independent except for the strong negative cross-covariance between the RL term and bandwidth error and aliasing. This is another evidence supporting the fact that MARL is able to mitigate bandwidth error and to a lesser extent the propagation of aliasing.

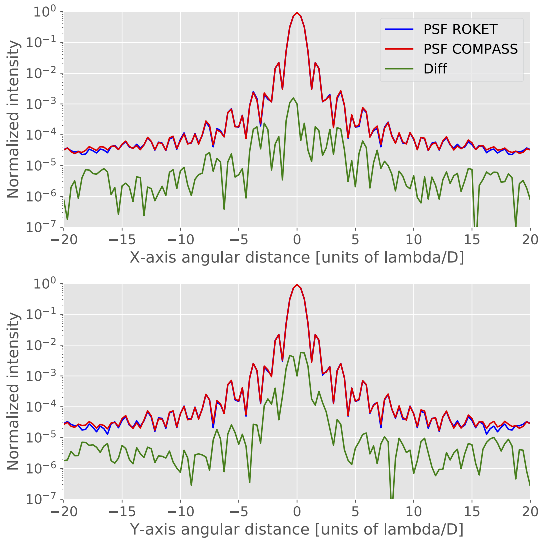


Fig. 9. COMPASS and ROKET reconstructions of the PSF with the MARL controller after 20000 frames. The green curve shows the difference in value between both PSF's. The L.E. SR is 0.918 in the case of ROKET and 0.919 in the case of COMPASS.

ROKET is also providing an error breakdown per mode, and we can thus compute the full error budget (i.e. the contribution from each error term) per mode. We used this feature to compute, in Fig. 10(b), an error breakdown per agent, each controlling a group of modes. As seen in this figure, the MARL controller provides a significant improvement for all the modes over the integrator controller. Nevertheless, the correction on lower order modes and tip-tilt has the larger impact on performance since the spatial power spectrum of an AO corrected residual phase follows a negative power law (aliasing excluded).

Trying to explain why we are seemingly able to reduce the aliasing contribution, let's invoke the well-known fact that aliasing produces an over-estimation of the high spatial frequencies, i.e.

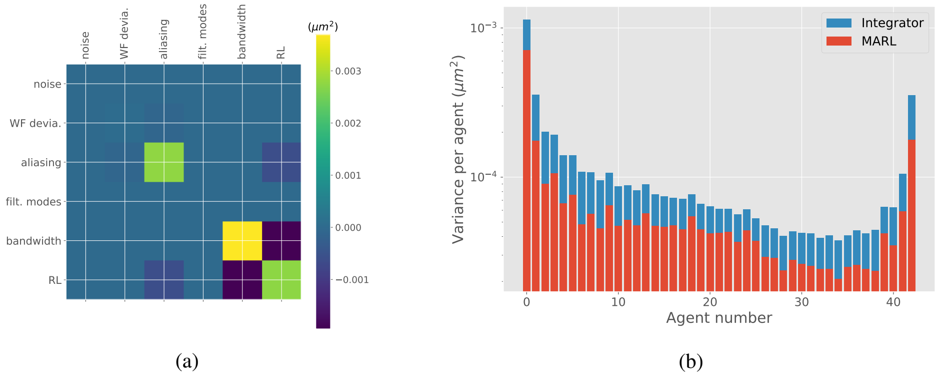


Fig. 10. a) Covariance of all the error terms in the current simulation: no noise, wind direction profile 1 and wind speed profile 2. b) Summed variance of modes controlled by a given agent (y axis in log-scale). The last agent controls tip-tilt. Other agents control 30 modes each and are ordered from lower order modes to higher order modes.

frequencies close to the cut-off frequency of the optical system (e.g. see Fig. 3 in [41]). While there is a good alignment between higher order B_{tt} modes and higher spatial frequencies, we hypothesise that since we are trying to minimise the power in the residual phase for all the modes, we are able to compensate a fraction of aliasing attributed to this overshooting in the higher order modes, as demonstrated in Fig. 10(b).

4.2.6. Comparison with the projection controller

In this section, we compare the commands produced for each frame against a controller based on the direct projection of the turbulent phase onto the DM without delay, which we call, projection controller. We use the simulation parameters and controllers used in the previous section.

We run the simulation for 1000 frames with the windowed MARL (without exploration), the integrator and the projection controllers and we compare the 1-norm distance of command vectors from both the integrator and MARL against the projection controller in the B_{tt} space. Figure 11 shows the sum of these distances for each B_{tt} mode for both controllers over the 1000 frames. Not surprisingly, since ROKET uses projection on B_{tt} modes as well, this figure leads to a similar conclusion as for Fig. 10(b) in previous section, that MARL performs better in most modes it controls.

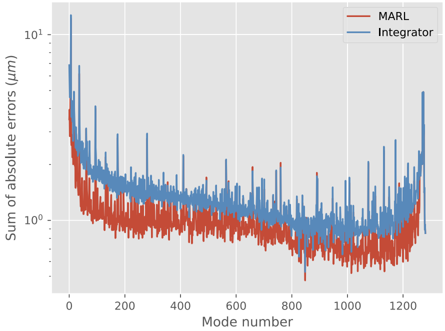
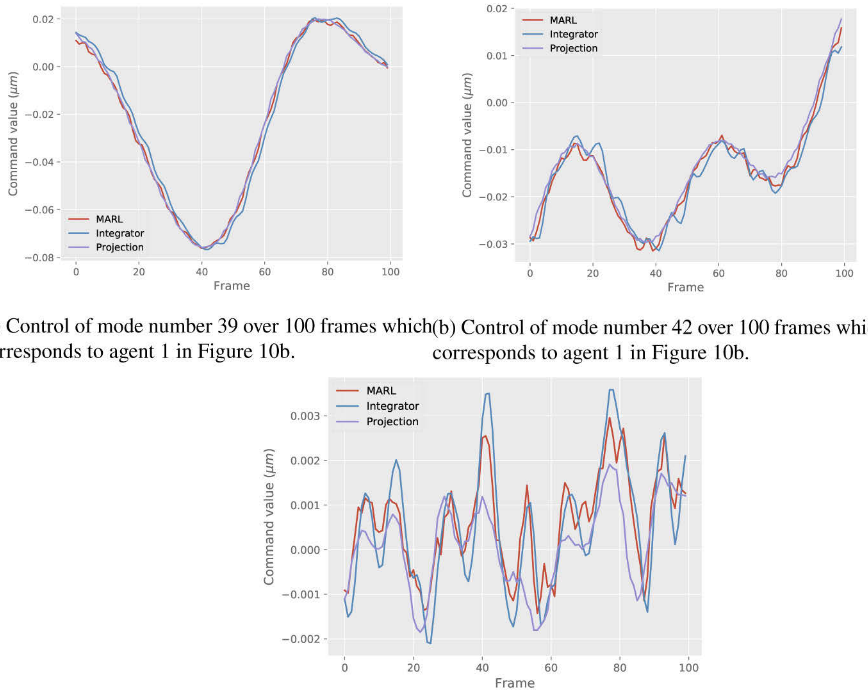


Fig. 11. Sum of the 1-norm distance between commands from both MARL and integrator controllers and the result of direct projection of the phase onto B_{tt} modes over 1000 frames. Tip-tilt is not included. As only 1262 modes are controlled through MARL, the last 21 modes are either controller by an integrator approach (16 modes) or discarded low sensitivity modes (5 modes) that is why both integrator and MARL have the same error on those.

Moreover, we can compare these commands at each timestep. Figure 12 shows such comparison for modes 39, 42 and 938, where we tried to choose modes exhibiting different overall behaviour. Comparing with the projection controller, we can observe different behaviour among modes. For example, mode 39, follows the projection curve perfectly, which could be interpreted as the MARL controller having learned a "model predictive behaviour". On the other hand, mode 42, shows a similar behaviour but with more variability around the projection solution. This could be due to intrinsic instability exhibited by the integrator on this mode, and, since the MARL controller is just offsetting the integrator controller commands, it may not be able to compensate the instability to its full extent. Finally, higher order modes, as shown for mode 938, show a more chaotic behaviour. Still, the MARL result seems closer to the projection controller than the integrator approach.

4.2.7. Comparison with LQG

In this section, we address question 7 above by comparing MARL against the latest implementation of LQG in COMPASS. LQG is a linear control scheme which is proven to be optimal when the system being controlled is linear, and both the process noise and measurement noise are



(a) Control of mode number 39 over 100 frames which corresponds to agent 1 in Figure 10b.

(b) Control of mode number 42 over 100 frames which corresponds to agent 1 in Figure 10b.

(c) Control of mode number 938 over 100 frames which corresponds to agent 31 in Figure 10b.

Fig. 12. Control of mode number 39 over 100 frames which corresponds to agent 1 in Fig. 10(b).

additive and jointly Gaussian [42]. Since AO is subject to both non-linear effects (e.g., WFS spot truncation), and non-Gaussian noise (e.g., Poisson-like photon noise), it is certainly possible that a given controller could match or outperform LQG. The LQG implementation, based on Correia [43], assumes frozen-flow turbulence evolution and requires *a priori* knowledge of the turbulence strength distribution and layer-wise wind-velocities. The specific LQG implementation is summarised in the first section of Cranney [44].

Table 3 shows the performance for each controller in different atmospheric conditions. We have experimented with atmospheric conditions with the same wind direction profile (direction profile 1) and different wind speeds profiles. The results shows similar performances for both LQG and MARL. However, this LQG implementation considers a perfect knowledge of the atmospheric parameters which may be affected by estimation errors in the context of an on-sky experiment. On the contrary, our RL approach is a data-driven method that can adapt online without any prior on the statistical properties of turbulence (e.g. being Kolmogorov) or corresponding parameters (e.g. number of layers, fraction of turbulence per layer, wind speed and direction profiles).

Table 3. L.E. SR of different controllers for 3 different seeds after 20000 frames. The best result for each atmospheric condition appears in bold. The MARL controller is the one that uses the windowed approach and the exploration is turned off.

Controller	v_1	v_2	v_3
Integrator	0.853 ± 0.0008	0.891 ± 0.0008	0.912 ± 0.0007
LQG	0.903 ± 0.0000	0.915 ± 0.0012	**0.924 ± 0.0008**
MARL (windowed)	**0.909 ± 0.0010**	**0.918 ± 0.0005**	0.922 ± 0.0007

4.2.8. Inference time analysis

In this section, we address question 8 on inference time analysis. We implemented both MARL and the autoencoder with the Pytorch [45] framework relying on the Python programming language. The server we used for these tests hosts an IBM Power9 8335-GTH CPU (40 cores) and a NVIDIA V100 GPU (16 GB).

As we did not parallelise the action selection for the different agents, we have measured the inference time for a single agent. We expect, once a parallelisation is implemented, to have similar inference times as the ones reported since the agents can work totally independently during the inference phase. Moreover, as explained in Section 3.3, SAC's exploration involves an extra computation step: sampling from a Gaussian distribution. This sampling can be turned off once we observe a convergence in performance.

Figure 13 provides the time-to-solution of inference for the autoencoder and a single agent of the MARL controller (with both exploration on and off). As observed in the figure above, both action selection and the autoencoder inference time are below 1 ms. Moreover, if we sum both the autoencoder and MARL inference times we are below the 2 *ms* threshold. While a non-perfect parallelisation of the MARL agents may increase slightly the inference times, we believe that optimising the controller implementation with more efficient frameworks such as TensorRT [46] and/or using specialised hardware such as field programmable gate arrays (FPGA) [47] may allow us to bring the inference time below 1 ms.

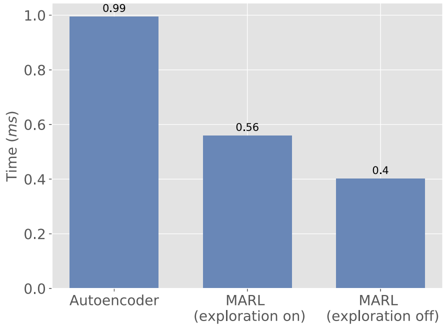


Fig. 13. Average inference time of all the components of the MARL controller for 1000 inferences in the closed-loop setting.

5. Discussion and future work

In this work, we have identified the different challenges of designing a RL method for AO control and proposed a solution with a Multi-Agent model-free Reinforcement Learning controller working as an additive corrector on top of an integrator controller with an autoencoder to mitigate the impact of noise. We showed that this solution, in a system composed of a SH-WFS with 40x40 subapertures installed on an 8m telescope, increases the performance over an integrator controller and has similar performance to *state-of-the-art* LQG controller with the added benefit of being a data driven method training online, during operations, without requiring any prior on the turbulence statistical properties and corresponding parameters.

Moreover, we have analysed the error budget with ROKET and have shown that the MARL controller is able to mitigate bandwidth error and to a lower extent the propagation of aliasing. We have also shown that the bandwidth error is addressed by learning a model predictive behaviour on some of the modes which, interestingly, is an emergent property of maximising the reward metric as we have not explicitly programmed the agent to be a model-based predictive controller.

Furthermore, we have shown that the advantage of our controller compared to similar ones such as the one proposed by [25] can be found in inference time. Time-to-solution for the combination of the denoising autoencoder and the MARL controller is below 2 ms considering perfectly parallelised architecture. However, future work should progress this aspect and deliver a final implementation that has a total inference time below 1 *ms*. A possibility regarding inference time, is discarding the autoencoder and implementing noise mitigation directly into the RL module. For instance, in some existing work on RL, authors have addressed noise by considering the worst case transition probability $P(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a})$ in what is known as robust Reinforcement Learning [48]. Another possible alternative to the autoencoder to tackle the effect of noise is to use convolutional layers in the MARL neural networks which may enhance the robustness in noisy settings (similar to the works of [14,25] among others). However, this may increase training/inference times for the MARL controller.

Concerning the turbulence setting, we have validated our method for different values of wind direction and wind speeds in a 3-layer profile configuration. We have also assessed the performance under changing turbulence conditions for which the MARL controller appears to be robust enough. On this front, future work must include more realistic transition between turbulence conditions with a more incremental evolution over time. Here again, to improve the robustness of MARL to changing atmospheric solutions, future work could look into few solutions already proposed by the RL community. As such, meta-learning (learning to learn) could be an approach to build upon, by adding an extra module on top of the proposed controller. For instance, in model-agnostic meta-learning [49], a policy is trained through multiple training tasks (in the case of AO we could train a policy for different values of r_0, wind speed and wind velocity) and then, tested on a set of evaluation tasks (in the case of AO this could be different atmospheric conditions not included in the training process) such that it would learn an effective policy on new unseen conditions through very few update steps.

Regarding training time, the current results show that it takes 17 minutes of operation to surpass the integrator controller's performance. We expect that finetuning the hyperparameters used for SAC will lower this amount of time. Moreover, the training time could also be greatly reduced by having pre-trained policies, using a realistic simulator such as COMPASS, for different atmospheric configurations and loading the one trained for similar conditions. In this case, training online will consist in a fine tuning step, which may be significantly faster than the current experiments, where training starts from scratch. In addition, as mentioned in the paper, we have designed the MARL solution working on top of the integrator controller. In a strategy using pre-trained policies and considering the results obtained with an unoptimised integrator controller, with low gain, using only RL without the initial integrator estimate could be viable and may lead to a better global optimum. Similarly, if implemented in an operational setting, the autoencoder would be trained previously on a bench simulating different guide star magnitudes and loading the one corresponding to the current operational scenario.

In regards to SAC exploration, where in the exploratory phase the agents choose non-optimal actions, we could either minimise the amount of exploration or turn it off completely once we observe that the performance increase plateaus and reactivate it if the atmospheric conditions change substantially.

One final concern for this family of controllers is stability. The so-called *catastrophic forgetting*, where a neural network performance drops to a great extent in a few update steps, may sometimes occur in Deep Reinforcement Learning. While SAC is notoriously more stable than other model-free methods, we do not discard this possibility. We may also consider Robust Reinforcement Learning not only to mitigate the effect of noise but also for stability purposes.

Another possible future work direction is to explore other WFS concepts such as the pyramid WFS. In the pyramid WFS non-linear errors have a stronger impact and the RL properties of learning non-linear policies may provide a significant advantage.

Finally, we plan on implementing such new approach based on RL on actual experiments, on the bench and possibly on-sky, in order to evaluate the performance and behaviour (e.g. training time, stability, changing atmospheric conditions) in a real setting and increase the readiness level of this technique for future extreme AO instruments.

The code for the project can be found in Ref. [50].

Funding. European Commission (871669, 873120).

Acknowledgments. We would like to thank Jesse Cranney for providing the LQG results.

Disclosures. The authors declare no conflicts of interest.

Data availability. This work generated data to train both the Multi-Agent Reinforcement Learning controller and the autoencoder. Reference [50] provides the code to create the training data. The Multi-Agent Reinforcement Learning data was generated on run-time and discarded but can be reproduced by setting the same seed as in the experiments. The datasets for the supervised training of the autoencoder were created offline. Due to its enormous size, the datasets are not explicitly provided, but the code to generate them is.

Additional data underlying the results presented in this paper that are not publicly available may be obtained from the authors upon reasonable request.

Supplemental document. See Supplement 1 for supporting content.

References

1. R. N. Paschall and D. J. Anderson, "Linear quadratic gaussian control of a deformable mirror adaptive optics system with time-delayed measurements," Appl. Opt. **32**(31), 6347–6358 (1993).
2. G. Sivo, C. Kulcsár, J.-M. Conan, H.-F. Raynaud, É. Gendron, A. Basden, F. Vidal, T. Morris, S. Meimon, C. Petit, D. Gratadour, O. Martin, Z. Hubert, A. Sevin, D. Perret, F. Chemla, G. Rousset, N. Dipper, G. Talbot, E. Younger, R. Myers, D. Henry, S. Todd, D. Atkinson, C. Dickson, and A. Longmore, "First on-sky scao validation of full lqg control with vibration mitigation on the canary pathfinder," Opt. Express **22**(19), 23565–23591 (2014).
3. G. Sivo, A. Turchi, E. Masciadri, A. Guesalaga, and B. Neichel, "Towards an automatic wind speed and direction profiler for wide field adaptive optics systems," Mon. Not. R. Astron. Soc. **476**(1), 999–1009 (2018).
4. B. Le Roux, J.-M. Conan, C. Kulcsár, H.-F. Raynaud, L. M. Mugnier, and T. Fusco, "Optimal control law for classical and multiconjugate adaptive optics," J. Opt. Soc. Am. A **21**(7), 1261–1276 (2004).
5. C. Petit, J.-M. Conan, C. Kulcsár, and H.-F. Raynaud, "Linear quadratic gaussian control for adaptive optics and multiconjugate adaptive optics: experimental and numerical analysis," J. Opt. Soc. Am. A **26**(6), 1307–1325 (2009).
6. C. Kulcsár, H.-F. Raynaud, C. Petit, J.-M. Conan, and P. V. De Lesegno, "Optimal control, observers and integrators in adaptive optics," Opt. Express **14**(17), 7464–7476 (2006).
7. P. Massioni, C. Kulcsár, H.-F. Raynaud, and J.-M. Conan, "Fast computation of an optimal controller for large-scale adaptive optics," J. Opt. Soc. Am. A **28**(11), 2298–2309 (2011).
8. O. Guyon and J. Males, "Adaptive optics predictive control with empirical orthogonal functions (eofs)," arXiv preprint arXiv:1707.00570 (2017).
9. D. A. Montera, B. M. Welsh, M. C. Roggemann, and D. W. Ruck, "Prediction of wave-front sensor slope measurements with artificial neural networks," Appl. Opt. **36**(3), 675–681 (1997).
10. H. Guo, N. Korablinova, Q. Ren, and J. Bille, "Wavefront reconstruction with artificial neural networks," Opt. Express **14**(14), 6456–6462 (2006).
11. R. Swanson, M. Lamb, C. Correia, S. Sivanandam, and K. Kutulakos, "Wavefront reconstruction and prediction with convolutional neural networks," Proc. SPIE **10703**, 107031F (2018).
12. X. Liu, T. Morris, C. Saunter, F. J. d. C. Juez, C. González-Gutiérrez, and L. Bardou, "Wavefront prediction using artificial neural networks for open-loop adaptive optics," Mon. Not. R. Astron. Soc. **496**(1), 456–464 (2020).
13. R. Landman and S. Haffert, "Nonlinear wavefront reconstruction with convolutional neural networks for fourier-based wavefront sensors," Opt. Express **28**(11), 16644–16657 (2020).
14. R. Swanson, M. Lamb, C. M. Correia, S. Sivanandam, and K. Kutulakos, "Closed loop predictive control of adaptive optics systems with convolutional neural networks," Mon. Not. R. Astron. Soc. **503**(2), 2944–2954 (2021).
15. R. Ragazzoni, "Pupil plane wavefront sensing with an oscillating prism," J. Mod. Opt. **43**(2), 289–293 (1996).
16. S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," The J. Mach. Learn. Res. **17**, 1334–1373 (2016).
17. H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, (2016), pp. 50–56.
18. M. Popova, O. Isayev, and A. Tropsha, "Deep reinforcement learning for de novo drug design," Sci. Adv. **4**(7), eaap7885 (2018).
19. B. Pou, E. Qui nones, D. Gratadour, and M. Martin, "Denoising wavefront sensor image with deep neural networks," Proc. SPIE **11448**, 114484J (2020).
20. I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT press, 2016).
21. X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder," in *Interspeech*, vol. 2013 (2013), pp. 436–440.

22. L. Gondara, “Medical image denoising using convolutional denoising autoencoders,” in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, (IEEE, 2016), pp. 241–246.
23. K. Hu, Z. Xu, W. Yang, and B. Xu, “Build the structure of wfsless ao system through deep reinforcement learning,” IEEE Photonics Technol. Lett. **30**(23), 2033–2036 (2018).
24. H. Ke, B. Xu, Z. Xu, L. Wen, P. Yang, S. Wang, and L. Dong, “Self-learning control for wavefront sensorless adaptive optics system through deep reinforcement learning,” Optik **178**, 785–793 (2019).
25. J. Nousiainen, C. Rajani, M. Kasper, and T. Helin, “Adaptive optics control using model-based reinforcement learning,” Opt. Express **29**(10), 15327–15344 (2021).
26. T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, “Benchmarking model-based reinforcement learning,” arXiv preprint arXiv:1907.02057 (2019).
27. R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction* (MIT press, 2018).
28. R. Landman, S. Y. Haffert, V. M. Radhakrishnan, and C. U. Keller, “Self-optimizing adaptive optics control with reinforcement learning for high-contrast imaging,” J. Astron. Telesc. Instruments, Syst. **7**(3), 039002 (2021).
29. 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,” arXiv preprint arXiv:1509.02971 (2015).
30. T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, “Soft actor-critic algorithms and applications,” arXiv preprint arXiv:1812.05905 (2018).
31. R. J. Noll, “Zernike polynomials and atmospheric turbulence,” J. Opt. Soc. Am. **66**(3), 207–211 (1976).
32. F. Ferreira, E. Gendron, G. Rousset, and D. Gratadour, “Numerical estimation of wavefront error breakdown in adaptive optics,” Astron. & Astrophys. **616**, A102 (2018).
33. G. Dulac-Arnold, D. Mankowitz, and T. Hester, “Challenges of real-world reinforcement learning,” arXiv preprint arXiv:1904.12901 (2019).
34. T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, “Residual policy learning,” arXiv preprint arXiv:1812.06298 (2018).
35. T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, “Residual reinforcement learning for robot control,” in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019), pp. 6023–6029.
36. A. Tavakoli, F. Pardo, and P. Kormushev, “Action branching architectures for deep reinforcement learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32 (2018), pp. 4131–4138.
37. F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs* (Springer, 2016).
38. R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman, “Solving transition independent decentralized markov decision processes,” J. Artif. Intell. Res. **22**, 423–455 (2004).
39. D. Gratadour, M. Puech, C. Vérinaud, P. Kestener, M. Gray, C. Petit, J. Brulé, Y. Clénet, F. Ferreira, E. Gendron, M. Lainé, A. Sevin, G. Rousset, F. Hammer, I. Jégouzo, M. Paillous, S. Taburet, Y. Yang, J.-L. Beuzit, A. Carlotti, M. Westphal, B. Epinat, M. Ferrari, T. Gautrais, J. C. Lambert, B. Neichel, and S. Rodionov, “Compass: an efficient, scalable and versatile numerical platform for the development of elt ao systems,” Proc. SPIE **9148**, 91486O (2014).
40. F. Roddier, “The effects of atmospheric turbulence in optical astronomy,” Progess in Optics **19**, 281–376 (1981).
41. C. Bond, T. Fusco, C. Correia, J. Veran, and J. Telxeira, “Anti-aliasing wave-front reconstruction with shack-hartmann sensors,” in *Adaptive Optics for Extremely Large Telescopes 4–Conference Proceedings*, vol. 1 (2015).
42. T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation* (Prentice Hall, 2000).
43. C. M. Correia, K. Jackson, J.-P. Véran, D. Andersen, O. Lardière, and C. Bradley, “Spatio-Angular Minimum-Variance Tomographic Controller for Multi-Object Adaptive-Optics Systems,” Appl. Opt. **54**(17), 5281–5290 (2015).
44. J. Cranney, J. De Doná, F. Rigaut, and V. Korkiakoski, “Zonal multi-layer predictive control for multi-conjugate adaptive optics: Mavis simulations,” *Adaptive Optics for Extremely Large Telescopes* (2019).
45. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, eds. (Curran Associates, Inc., 2019), pp. 8024–8035.
46. “NVIDIA TensorRT,” https://developer.nvidia.com/tensorrt. Retrieved 2021–08.
47. R. Wu, X. Guo, J. Du, and J. Li, “Accelerating neural network inference on fpga-based platforms–a survey,” Electronics **10**(9), 1025 (2021).
48. J. Morimoto and K. Doya, “Robust reinforcement learning,” Neural computation **17**(2), 335–359 (2005).
49. C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *International Conference on Machine Learning*, (PMLR, 2017), pp. 1126–1135.
50. B. Pou, E. Quinones, D. Gratadour, and M. Martin, “Implementation of Adaptive Optics control with Multi-Agent Model-Free Reinforcement Learning,” Github (retrieved 2021-12), https://github.com/Tomeu7/AO-MARL.