Self-optimizing adaptive optics control with reinforcement learning for high-contrast imaging

Landman, R.; Haffert, S.Y.; Radhakrishnan, V.M.; Keller, C.U.

Self-optimizing adaptive optics control with Reinforcement Learning for high-contrast imaging

Rico Landman[a,*], **Sebastiaan Y. Haffert**[a,b,c], **Vikram M. Radhakrishnan**[a], **Christoph U. Keller**[a]

[a]Leiden Observatory, Leiden University, PO Box 9513, 2300 RA Leiden, The Netherlands
[b]Steward Observatory, Unversity of Arizona, 933 North Cherry Avenue, Tucson, Arizona
[c]NASA Hubble Fellow

Abstract. Current and future high-contrast imaging instruments require extreme adaptive optics (XAO) systems to reach contrasts necessary to directly image exoplanets. Telescope vibrations and the temporal error induced by the latency of the control loop limit the performance of these systems. One way to reduce these effects is to use predictive control. We describe how model-free Reinforcement Learning can be used to optimize a Recurrent Neural Network controller for closed-loop predictive control. First, we verify our proposed approach for tip-tilt control in simulations and a lab setup. The results show that this algorithm can effectively learn to mitigate vibrations and reduce the residuals for power-law input turbulence as compared to an optimal gain integrator. We also show that the controller can learn to minimize random vibrations without requiring online updating of the control law. Next, we show in simulations that our algorithm can also be applied to the control of a high-order deformable mirror. We demonstrate that our controller can provide two orders of magnitude improvement in contrast at small separations under stationary turbulence. Furthermore, we show more than an order of magnitude improvement in contrast for different wind velocities and directions without requiring online updating of the control law.

Keywords: Adaptive Optics, Predictive Control, High Contrast Imaging, Reinforcement Learning, Machine Learning.

*Rico Landman, rlandman@strw.leidenuniv.nl

1 Introduction

One of the main limitations of current and future ground-based high contrast imaging (HCI) instruments is the wavefront error induced by the time lag between sensing and correcting the wavefront.[1] This time lag results in a halo of speckles along the wind direction, which limits the contrast at small separations.[2] Furthermore, the finite control bandwidth results in low-order residuals after the correction, such as tip and tilt. One of the main causes of these residuals are vibrations close to the maximum control frequency. All ground-based high-contrast imaging instruments suffer from vibrations, including SCExAO,[3] GPI[4] and SPHERE.[5] These vibrations decrease the resolution of long-exposure images. Furthermore, some coronagraphs are highly sensitive to residual tip-tilt,[6,7] which will further reduce the contrast.

Both of these effects can be reduced with better control algorithms. Early work on improved controllers involved optimizing the modal gains of a modal integrator based on the open-loop temporal Power Spectral Density (PSD).[8,9] This optimal gain is a trade-off between turbulence rejection and the amplification of noise and disturbances outside the control bandwidth. However, to reduce the effect of the servo-lag we have to predict the disturbances before they are measured. A popular approach to optimal control is Linear Quadratic Gaussian (LQG) control,[10] which is based on a Kalman filter. Variants of LQG-based controllers have been demonstrated to reduce tip-tilt vibrations in simulations,[11] lab verifications[12] and on-sky.[4,13,14] A disadvantage of LQG-like controllers is their reliance on a linear state-space model of the turbulence and deformable

mirror (DM) dynamics;[15] this requires accurate calibration of the model parameters. Hence, these controllers can also not correct for correlations in the turbulence that are not included in the model. A combination of an LQG controller for low-order vibrations and an optimal modal gain integrator for high-order modes is currently being used in the adaptive optics (AO) systems of both SPHERE[14] and GPI.[16]

Another group of data-driven algorithms construct a linear filter that minimizes the residual phase variance from (pseudo) open-loop data. This idea was originally proposed and demonstrated for modal control by Dessenne et al.[17] More recently, a spatio-temporal filter based on Empirical Orthogonal Functions (EOFs) was demonstrated to significantly increase the contrast in a simulated HCI system.[18] The performance of such linear filters was tested on open-loop AO telemetry for the Keck II AO system[19] and SPHERE[20] and showed a decrease in residual wavefront error. An issue with these data-driven predictive control algorithms is that they rely on the prediction of (pseudo) open-loop wavefront data, while XAO systems operate in closed-loop. To use these controllers, one has to reconstruct the pseudo open-loop residuals. Not only does this require knowledge of the servo-lag of the system and DM dynamics, it is also not trivial in the case where the response of the wavefront sensor is nonlinear, such as for the Pyramid Wavefront sensor.[21] To circumvent the issue of nonlinearity, neural networks can improve the performance compared to linear models when using an unmodulated Pyramid Wavefront Sensor.[22] However, predicting the open-loop residuals alone does not provide the optimal closed-loop control law. For example, one still has to find the best gain for the control loop. Furthermore, as the open-loop wavefront may be orders of magnitude larger than the closed-loop errors, a relative error in the predicted open-loop wavefront may be detrimental to the closed-loop contrast as compared with a relative error in the closed-loop residuals. Obtaining a closed-loop control law using supervised learning approaches is hampered by the distribution of inputs depending on the control law itself. It was recently shown that this can be mitigated by using an adversarial prior in the training of a neural-network controller.[23] Instead of predicting in open-loop while controlling in closed-loop, we can also do data-driven predictive control directly in closed-loop. This can for example be done using data-driven subspace predictive control, which was recently shown to provide orders of magnitude improvement in contrast in simulations and in the lab.[24] This does not require the explicit prediction of the wavefront after a known servo-lag but directly optimizes the closed-loop commands. The proposed approach here is similar, but generalizes to nonlinear control laws and arbitrary optimization objectives.

Another issue for most of these approaches is the non-stationary nature of turbulence.[25] All methods mentioned above require online updating of the control parameters based on the current turbulence parameters. This updating of the control law is computationally very expensive and often involves inverting large matrices.[18,26] Recently, it was shown that a single Artificial Neural Network (ANN) can learn to predict the open-loop wavefront for varying wind speeds and directions for a simulated AO system.[27] This eliminates the need for updating the control law to keep up with changing turbulence properties. This would allow us to decouple the training of the controller from online control, simplifying the implementation and potentially gaining in stability. However, using this approach still leaves the problems associated with predicting the open-loop turbulence.

Here, we use Reinforcement Learning (RL), a Machine Learning approach, for the data-driven optimization of the closed-loop adaptive optics control law. Indeed, Machine Learning has recently received significant attention in the AO community for wavefront reconstruction,[28,29] prediction[27,30] and control;[22,23] Reinforcement Learning was already suggested as a method to directly

optimize the contrast in a high-contrast imaging system[31] and for focal-plane wavefront control.[32] Very recently, model-based RL was proposed for adaptive optics control,[33] and was shown to reduce the temporal error and adjust to mis-registrations between the deformable mirror and wavefront sensor. Our proposed approach here has similar advantages as their approach. The main difference is the use of a parameterized control law here over a planning algorithm. This avoids the need for solving the planning optimization for every control command, which is generally too computationally expensive to do at kHz frequencies, at which typical XAO systems are run. However, this generally comes at the cost of longer training times.

Our RL-based approach has some distinct advantages:

- **Data-driven**: The algorithm is completely self-optimizing and data-driven; it does not require an explicit model of the disturbances or system dynamics, or knowledge of system parameters. Instead of explicitly predicting the wavefront after an assumed servo-lag, it directly optimizes the closed-loop commands.
- **Flexible**: The algorithm allows for an arbitrary optimization objective. The user is free to choose the metric to optimize based on the science goals of the instrument. Furthermore, it can handle multiple inputs including systems with multiple wavefront sensors.
- **Nonlinear**: The proposed method can optimize any parameterized, differentiable control law. For example, one can optimize the parameters of a nonlinear Artificial Neural Network (ANN) controller. This allows the controller to model the nonlinear dynamics of the system, such as for systems with a Pyramid Wavefront Sensor. Because of the larger representation power of these Neural Networks, they can also learn to perform under varying atmospheric conditions.

This paper is structured as follows: Section 2 provides an introduction to Reinforcement Learning and describes the relevant algorithm. Section 3 simulates the algorithm's performance in tip-tilt control for vibrations and a power-law input, while section 4 shows the results from a lab experiment. Section 5 shows the results of using the algorithm to control a high-order deformable mirror. Finally, section 6 draws conclusions and outlines future work.

2 Reinforcement Learning Control

We begin the description of our algorithm with an introduction to the Reinforcement Learning framework, within which we formulate the AO control problem. This is followed by a discussion of the Deterministic Policy Gradient algorithm and how it can be applied to closed-loop AO control.

2.1 AO control as a Markov Decision Process

Reinforcement Learning[34] (RL) algorithms deal with Markov Decision Processes (MDPs), discrete-time stochastic control processes. In the RL framework, an agent operates within an environment, which is formally defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$ and time index t. Here, $s_t \in \mathcal{S}$ is the state of the environment, $a_t \in \mathcal{A}$ the action taken by the agent, $\mathcal{R}$ the reward function and $\mathcal{P}$ the state transition probabilities defined by:

$$\mathcal{P}(s_t, a_t, s_{t+1}) = P(s_{t+1}|s_t, a_t), \tag{1}$$

Table 1 Overview and explanation of the various terms used in the Reinforcement Learning (RL) framework.

Symbol	RL term	AO term/Explanation
s	State	Input to the controller.
o	Observation	Measurement of the residual wavefront
a	Action	Incremental DM commands
r	Reward	Measure of the instantaneous performance; we are free to choose this
R	Return	Discounted sum of future rewards; the optimization objective
π_θ	Actor/policy	The control law
Q_ω	Critic	Estimate of the return/the cost function

where $P(s_{t+1}|s_t, a_t)$ gives the probability of transitioning from state s_t to state s_{t+1} as a result of action a_t. The state needs to be defined in such a way that the transition probabilities are fully described by Eq. 1 and do not depend on previous states (e.g. s_{t-1}). This is also known as the Markov property.

In the case of AO control, the agent is the DM controller. The environment consists of everything but the controller, including the evolution of the turbulence and the DM dynamics. At each time step the the controller receives a state s_t. For example, in the case of an integral controller the state consists of only the most recent observation of the residual wavefront o_t. The representation of the state for our Reinforcement Learning controller will be extensively discussed in section 2.4. Based on this state, the controller takes an action a_t, following some control law π: $a_t = \pi(s_t)$. Since the controller operates in closed loop, this action consists of the incremental voltages added to the DM, equivalent to an integral controller. The DM commands and the temporal evolution of the environment result in the transition to a new state s_{t+1}, according to the transition probabilities $\mathcal{P}$ of the stochastic process of the turbulence, and a reward $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$. This reward is a measure of the performance of the controller; the reward function $\mathcal{R}$ can be arbitrarily chosen as long as it can be calculated at every time step. The goal is to determine the control law π that gives the highest cumulative future reward. To ensure that the cumulative future reward remains finite and to prioritize immediate rewards, future rewards are discounted at a rate $\gamma < 1$, the discount factor. The discounted future return R_t is then defined as:

$$R_t = \sum_{t=0} \gamma^t r_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \ . \tag{2}$$

The optimization objective can then be formally defined as:

$$J(\pi) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi}[R]. \tag{3}$$

Here, $\mathbb{E}$ denotes the expectation value and ρ^π the state visitation distribution when following the control law π. This state visitation distribution describes how often we end up in state s if we follow the control law π. This expectation over the state visitation distribution is taken because we want to optimize over the states that we observe. It also accounts for the dependency of the inputs on the controller. In the rest of this work, when we denote an expectation value, it is always over $s \sim \rho^\pi$ and $a \sim \pi$, and we will leave this out for readability. There are a variety of algorithms to find an optimal control law in the above framework.[35] One of the most frequently used algorithms for continuous control problems is the (Deep) Deterministic Policy Gradient algorithm,[36,37] which

we will use here. Our choice for the DDPG algorithm was motivated by the need for a directly parameterized and deterministic control law, which allows for the fast computation of the DM commands at kHz frequencies without needing to solve a planning optimization problem every millisecond. Furthermore, while not explored in this work, we preferred an algorithm which allows for an arbitrary optimization objective and could therefore also be used for the direct optimization of a focal plane quantity, such as the Strehl ratio or the post-coronagraphic contrast. Finally, since it is generally easy to collect a lot of data, as we generate 1000 datapoints per second at 1 kHz, the lower sample efficiency of model-free[1] algorithms was considered less of an issue.

2.2 (Deep) Deterministic Policy Gradient

The (Deep) Deterministic Policy Gradient (DPG)[36] algorithm uses two parameterized models:

- An **Actor** $\pi_\theta(s)$ that models the control law with parameters θ. This model maps the state to the DM commands : $a_t = \pi_\theta(s_t)$.
- A **Critic** $Q_\omega(s, a)$ that models the expected return for a given state and DM command with parameters ω: $Q_\omega(s_t, a_t) = \mathbb{E}[R_t|s_t, a_t]$. This is an estimation of the cost function that has to be optimized.

Both the actor and critic may be any kind of parameterized, differentiable model. These may for example be function approximators, such as Artificial Neural Networks (ANN).[38] The DPG algorithm with ANNs as function approximators is referred to as the Deep Deterministic Policy Gradient (DDPG) algorithm.[37] In this case, θ and ω are the trainable parameters of the ANN for the actor and critic, respectively. The algorithm aims to find the parameters θ of the controller that maximizes the optimization objective $J(\pi_\theta)$. Using the actor and the critic we can calculate the gradients of the optimization objective with respect to the control parameters using the chain rule:

$$\begin{aligned}\nabla_\theta J(\pi_\theta) =& \mathbb{E}[\nabla_\theta Q_\omega(s, a)] \\ =& \mathbb{E}[\nabla_a Q_\omega(s, a) \nabla_\theta \pi_\theta(s)]\end{aligned} \tag{4}$$

These gradients can then be used by a gradient-based optimizer to update the parameters θ of the control law in the direction in which the expected return increases. The quality of the controller thus inherently depends on the ability of the critic $Q_\omega(s, a)$ to successfully model the expected return. The critic can be taught using Temporal Difference (TD) learning, which uses the Bellman equation[39] to bootstrap the expected future reward:

$$\begin{aligned}Q_\omega(s_t, a_t) =& \mathbb{E}[R_t|s_t, a_t] \\ =& \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots] \\ =& \mathbb{E}[r_t + \gamma(r_{t+1} + \gamma r_{t+2} + \dots)] \\ =& \mathbb{E}[r_t + \gamma Q_\omega(s_{t+1}, \pi(s_{t+1}))] \equiv \mathbb{E}[y_t],\end{aligned} \tag{5}$$

where y_t are referred to as the target values and are calculated using:

$$y_t = r_t + \gamma Q_\omega(s_{t+1}, \pi(s_{t+1})). \tag{6}$$

[1] Model-free in the RL context does not imply that we do not have an empirical model for the controller. Instead, it means that the algorithm does not explicitly model the transition dynamics and plans through this model to find the optimal commands/policy.

Training the critic can then be framed into a supervised learning problem for each iteration, where we try to find the parameters ω that minimize the mean-squared error between the output of the critic and the target values y_t. The critic loss L is thus given by:

$$L(\omega) = -\frac{1}{2}\mathbb{E}[(y_t - Q_\omega(s_t, a_t))^2], \tag{7}$$

and its gradients with respect to the parameters ω by:

$$\nabla_\omega L(\omega) = -\mathbb{E}[(y_t - Q_\omega(s_t, a_t))\nabla_\omega Q_\omega(s_t, a_t)]. \tag{8}$$

We can then use a gradient-based optimizer to update the parameters in the direction that minimizes the loss. However, since the target values depend on the critic itself, this may quickly lead to instabilities in the training process. It is therefore common to use target models Q' and π' to calculate the target values y_t.[37] The parameters of these target models slightly lag behind the true models. For every iteration i of training the critic, the target models are updated as:

$$\begin{aligned} \theta'_i &= \tau\theta_i + (1-\tau)\theta'_{i-1} \\ \omega'_i &= \tau\omega_i + (1-\tau)\omega'_{i-1}. \end{aligned} \tag{9}$$

Here, $\tau < 1$ is a hyper-parameter that determines how quickly the target models are updated.

2.3 Data collection

Updating the actor and critic requires evaluating expectation values over the state visitation distribution. This can naturally be done by collecting tuples (s_t, a_t, r_t s_{t+1}) while following the control law π_θ. For our purpose, this is done by running the AO system in closed-loop with the controller π_θ. This data collection process is often separated into episodes of finite length, after which the sequence of tuples is saved in a so-called replay buffer, which is our training data set. For computational reasons, the expectation values are often estimated over a limited number of these tuples, a batch, instead of over the full data set. To reduce the correlation in a batch, the tuples are randomly sampled from the replay buffer.

The DDPG algorithm is off-policy,[36,37] meaning that it is not strictly required to sample the training data with the current policy. In principle, we can use any other controller, or even random actions, to generate the training tuples of states, actions and rewards. While this introduces a mismatch between the distribution of the training data and the true state visitation distribution of the current controller, this is often ignored in practice. Therefore, we can use previously collected data, where the control law was different, to train the controller. Furthermore, it is also possible to include data from a completely different controller, such as an integrator, to train the RL controller. This way, the algorithm could also make use of historically collected telemetry data using the current controller at the telescope.

Because the algorithm allows for the data collection to be done with a different controller, we can also add some known disturbances to the commands. This is called exploration in the RL framework. For example, we can add some zero mean, normally distributed exploration noise with standard deviation σ:

$$a_t = \pi_\theta(s_t) + \mathcal{N}(0, \sigma) \tag{10}$$

This results in the controller not always giving the same DM commands for a given state, such that we can observe the results of these different commands. A common trick is to use large initial exploration noise and slowly decay it. This allows the algorithm to first observe the results of large changes in DM commands and finally observe the result of small changes in these commands. We empirically choose the standard deviation of the exploration noise during episode k as:

$$\sigma_k = \frac{\sigma_0}{1 + \zeta k} \text{ rad.} \tag{11}$$

Here, σ_0 is a hyper-parameter that specifies the initial standard deviation of the exploration noise and ζ how quickly it decays.

2.4 State representation

It is crucial to have a representation of the state s_t such that the problem obeys the Markov property (Eq. 1), which requires that the state-transition probabilities are fully described by the current state and the action taken. This is not trivial in closed-loop AO control. A first guess for the state might be the most recent wavefront observation, o_t. This may, for example, be retrieved from a dedicated wavefront sensor (WFS). However, defining the state like this is not consistent with the Markov property. First of all, we do not have access to the full state of the environment but only observe a noisy representation with the WFS measurements. Furthermore, the most recent wavefront measurement does not contain all the required information; for example, it lacks information about possible vibrations and wind flow. When the AO loop is closed, we also have to distinguish changes in the turbulence profile from changes due to previous commands, as we only observe the closed-loop residuals. These issues may be solved using state augmentation, where sequences of previous wavefront measurements and applied DM commands are used in the state. However, if we want our controller to be able to perform under varying conditions, it may require a large number of previous measurements to have a full representation of the state. For example, Guyon & Males[18] used a vector of the previous 800 measurements to correct tip-tilt vibrations. A disadvantage of this is that it increases the computational demand. Furthermore, all previous measurements are in principle weighted equally, which may lead to additional noise if the problem is not correctly regularized.

To circumvent the problem of partial observability, it was shown that Recurrent Neural Networks (RNNs) are able to solve control problems even for partially observable Markov Decision Processes, where the Markov property is not obeyed.[40,41] RNNs are a type of neural network often used in processing sequential data.[38,42] Instead of only mapping an input to an output, RNNs also update a memory state m_t based on the previous memory state and the input. This allows RNNs to summarize previous information in this memory state and use it to calculate the output. This enables RNNs to learn an internal representation of the true state of the system based on sequences of noisy measurements. As an additional advantage calculating the control commands only requires the most recent observation to be propagated through the controller. This is computationally less demanding, which may be important for high-order AO systems operating at kHz frequencies. Furthermore, RNNs also effectively use the temporal structure of the data by incorporating basic priors, such as that the most recent measurement is the most relevant instead of using one large vector of components with equal weights. In our case, we will use a Long Short-Term Memory (LSTM) cell[43] as the recurrent architecture. Training is now done on sequences of states, actions

and rewards using Truncated Backpropagation Trough Time (TBTT) over a length l. This version of the algorithm is known as the Recurrent Deterministic Policy Gradient (RDPG) algorithm.[41]

The state for closed-loop AO now consists of only the most recent observation of the wavefront o_t and the previous DM commands a_{t-1}:

$$s_t = (o_t, a_{t-1}) \ . \tag{12}$$

2.5 Algorithm Overview

The algorithm can be summarized in three main steps that are iteratively performed:

1. Collect training sequences $(o_1, a_1, r_1, \ldots o_t, a_t, r_t)$ by running in closed loop and storing the data in the replay buffer.
2. Train the critic using Temporal Difference learning on the observed data.
3. Improve the control law using the gradients obtained from the critic.

The pseudo-code describing the full algorithm can be found in Algorithm 1, and a schematic overview is shown in Fig 1.

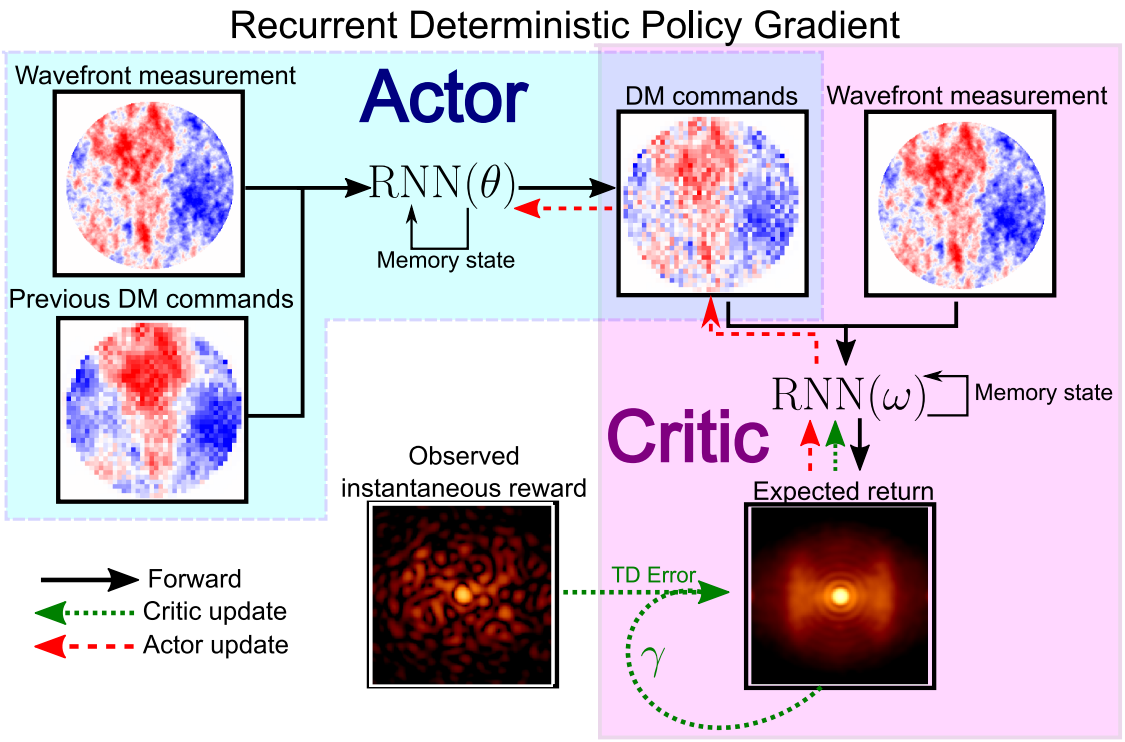

Fig 1 Visual overview of the algorithm. The black line indicates forward propagation. The green and red lines indicate how we backpropagate the gradients to update the critic and the actor, respectively.

Algorithm 1: Recurrent Deterministic Policy Gradient for closed-loop AO (based on Heess et al.[41])

Initialize critic parameters ω and actor parameters θ randomly
Initialize target network parameters $\omega' = \omega$, $\theta' = \theta$
Initialize replay buffer
Initialize σ_0 and ζ
for *k=1,number of episodes* **do**
  Reset controller memory states $m_t = 0$ and deformable mirror
  for *t=1,T* **do**
    Receive wavefront measurement o_t, reward r_t, construct state $s_t = (o_t, a_{t-1})$ and select incremental DM commands $a_t = \pi(s_t) + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma_k)$
  end
  Save trajectories $(o_1, a_1, r_1, \ldots, o_T, a_T, r_T)$ in replay buffer
  Decay exploration noise $\sigma_k = \frac{\sigma_0}{1+\zeta k}$
  for *j=1,number of training steps* **do**
    Randomly sample N batches of length l from the replay buffer
    For each batch i construct state histories $h_t^i = ((o_1^i, a_0^i) \ldots, (o_t^i, a_{t-1}^i))$
    Calculate the critic targets using the Bellman equation:

$$y_t^i = r_t^i + \gamma Q'_{\omega'}(h_{t+1}^i, \pi'_{\theta'}(h_{t+1}^i))$$

    Calculate the sampled critic loss gradient with TBTT:

$$\nabla_\omega L(\omega) \approx \frac{1}{Nl} \sum_{i=1}^{N} \sum_{t=1}^{l} (y_t^i - Q_\omega(h_t^i, a_t^i)) \nabla_\omega Q_\omega(s_t^i, a_t^i)$$

    Calculate the sampled Policy Gradient with TBTT:

$$\nabla_\theta J(\theta) \approx \frac{1}{Nl} \sum_{i}^{N} \sum_{t}^{l} \nabla_a Q_\omega(h_t^i, a_t^i) \nabla_\theta \pi_\theta(h_t^i)$$

    Use a gradient-based optimizer (e.g. Adam) to update the actor parameters θ and critic parameters ω.
    Update the target network parameters:

$$\theta'_j = \tau\theta_j + (1-\tau)\theta'_{j-1}$$
$$\omega'_j = \tau\omega_j + (1-\tau)\omega'_{j-1}$$

  end
end

3 Tip-tilt control in simulations

3.1 Simulation setup

As a proof of concept we apply the algorithm to tip-tilt control in simulations. We simulate an idealized optical system using the HCIPy[44] package for python. Our system has an unobscured, circular aperture of 1 meter in diameter and operates at a wavelength of 1 μm. The simulated deformable mirror has only two degrees of freedom, its tip and tilt. The observation o_t is the center of gravity (x_t, y_t) of the focal plane image in units of λ/D. The measured center of gravity along with the previous DM commands a_{t-1} are fed into a controller that controls tip and tilt in closed loop. To represent the servo-lag of estimating the wavefront tip and tilt, calculating and applying the control commands, we delay the DM commands by a discrete number of frames:

$$\mathrm{DM}_t = \mathrm{DM}_{t-1} + a_{t-\tau}, \tag{13}$$

where τ is the servo lag. This is equivalent to the controller seeing the state from τ iterations ago, which implies a delay in obtaining the reward. Throughout this section we assume an AO system operating at 1 kHz with a servo lag of 3 frames and do not consider any detector noise. Furthermore, we only simulate tip-tilt errors and ignore all higher order modes in the simulations.

3.1.1 Algorithm setup

To minimize the residual tip-tilt jitter we choose the squared deviation of the center of gravity of the PSF as the reward function:

$$r_t = -\frac{x_t^2 + y_t^2}{b^2}. \tag{14}$$

Here, b is a scaling factor for which we use a value of $4\lambda/D$ in the simulations. This scaling factor acts as a normalization of the rewards. Here, we empirically choose a value for the scaling factor, but this could also be inferred from the data. Important to note is that this scaling factor also scales the gradient of the reward with respect to the DM commands and therefore also influences the gradient update of the actor.

We use the same Neural Network architecture for both the actor and the critic. The input consists of the wavefront measurement $o_t = (x_t, y_t)$ and the previously applied tip-tilt commands $a_t = (a_{x,t-1}, a_{y,t-1})$. We use a Long Short-Term Memory[43] (LSTM) cell with 64 neurons as the recurrent layer in both the actor and the critic. For the critic the DM commands a_t are appended to the output of the LSTM. After that, we have a fully connected layer with 64 neurons for both networks. Finally, the output of the actor consists of two neurons, the additional tip and tilt commands, while the output of the critic is a single neuron that estimates the expected future return $Q(s_t, a_t)$. The LSTM uses a tanh activation function for the input and output gates and a hard sigmoid for the recurrent gates while the fully connected layer uses the ReLU activation function. The output of the actor again uses a tanh activation function to constrain the incremental DM commands between -1 rad and +1 rad. The output of the critic has a linear activation function. The architectures are also listed in Table 2. Both the actor and critic have $\sim 22,000$ free parameters. This is likely a lot more than required for tip-tilt control. Optimization of the architecture may result in shorter training times and improved performance, but is beyond the scope of this work.

We use the gradient-based Adam optimizer algorithm[45] with default parameters except for the learning rate. After every 500 iterations (an episode), we reset the DM shape. To reduce the

correlation of the data in a batch early on, we only start training after a certain number of episodes, the warmup. We use a warmup of 5 episodes, which is equivalent to 2500 iterations, and the exploration law given in Eq. 11. The hyperparameters of the RDPG algorithm are given in Table 3. The algorithm and Neural Networks are implemented using a combination of the TensorFlow[46] version 1.15 and Keras[47] packages for Python.

Table 2 Neural Network architectures of the actor and critic for the tip-tilt control.

	Layer type	Neurons	Input shape	Activation function
Actor	LSTM	64	(4)	Tanh
	Dense	64	(64)	ReLU
	Dense	2	(64)	Tanh
	Layer type	Neurons	Input shape	Activation function
Critic	LSTM	64	(4)	Tanh
	Concatenate	-	(64) and (2)	-
	Dense	64	(66)	ReLU
	Dense	1	(64)	Linear

Table 3 Hyperparameters

Parameter	Value
Actor learning rate	10^{-5}
Critic learning rate	10^{-3}
Target network soft update τ	10^{-3}
Discount factor γ	0.99
Batch size	64
TBTT length l	50 ms
Initial exploration σ_0	0.3 rad
Exploration decay ζ	0.005
Episode length	500 iterations
Number of training steps per episode	500

3.2 Vibration Suppression

As an example we consider an input disturbance consisting of three pure vibrations along each of the x and y directions. Along the x direction we have vibrations with frequencies at 13 Hz, 37 Hz and 91 Hz and along the y direction at 11 Hz, 43 Hz and 87 Hz. The gain of the integrator is optimized by running in closed loop for 1 second and choosing the gain that gives the lowest residual root mean square (RMS) center deviation:

$$\text{RMS} = \sqrt{< x_t^2 + y_t^2 >}. \tag{15}$$

Figure 2 shows the evolution of the RMS during the training of the RL controller. It shows that after a few thousand iterations the RL controller outperforms the integrator and eventually reaches an average RMS that is a factor 6 lower than for the integrator. The figure also shows the temporal PSD of the residuals along the x-direction for the fully trained controller without exploration

noise. This PSD is estimated using Welch's method[48] for a simulation of 10 seconds. The integrator reduces the power of the vibration at 13 Hz, but the other vibrations fall outside its correction bandwidth, even adding power to the 91 Hz vibration. The RL controller almost completely removes the power from all three vibrations. Most remaining power is at a frequency of 13 Hz. This may be explained by the fact that the period of this vibration is longer than the optimization length of 50 ms. The residuals in the x-direction for a 500-ms simulation for the RL controller, integrator and without correction are shown in Fig. 2. We see that after a short time of initializing the memory states, the RL controller is able to mitigate the vibrations.

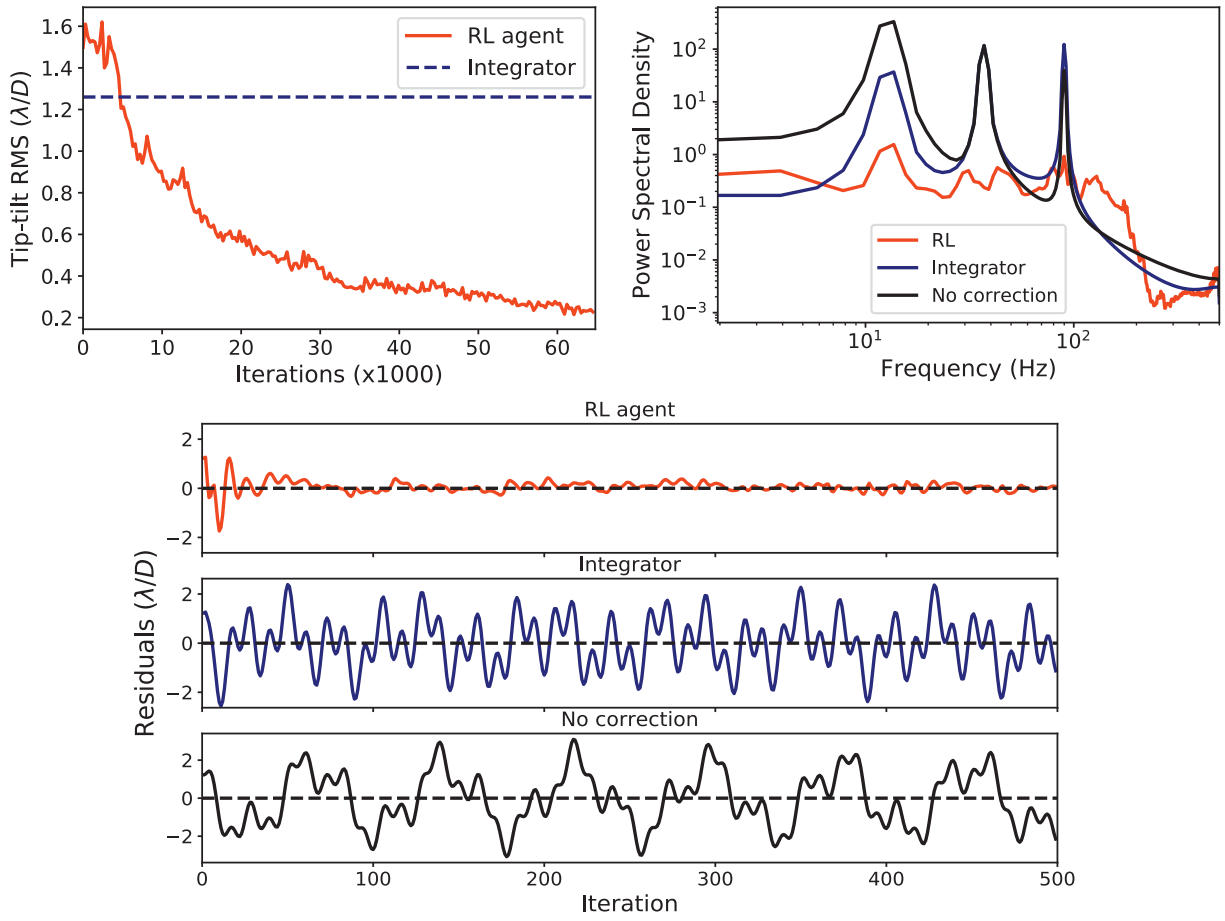

Fig 2 Top left: Training curve of the Reinforcement Learning controller under input disturbance consisting of three vibrations along both directions. Also shown is the average performance of an integrator with optimal gain. **Top right:** Temporal PSD of the residuals in the x-direction for a simulation of 10 seconds. Note that the lines of the input spectrum and integrator overlap at the peak at 37 Hz. **Bottom:** Residuals for the different controllers in the x-direction for a 500 ms simulation.

3.3 Power-law Disturbances

Next, we consider input disturbances following a temporal power law with a power-law index of -8/3 along both directions. We chose this power-law index because phase fluctuations for Kolmogorov turbulence are described by a -8/3 power-law.[49] For every episode, we generate random

power-law time series for 500 ms, neglecting frequencies below 2 Hz. The training curve is shown in Fig. 3. We again observe better RMS performance for the RL controller as compared to an integrator with optimized gain. We test the trained controller by running in closed loop for 10 s and calculating the temporal PSD of the residuals, which are also shown in Fig. 3. The RL controller has a larger rejection power at lower frequencies. The residual PSD of the integrator exhibits a bump around 70 Hz, which is the result of the integrator overshooting. A lower gain would reduce this overshooting but would also decrease its rejection power at lower frequencies. The RL controller can account for already applied commands that are not yet visible in the residual measurements due to the servo lag, allowing it to use a higher gain without overshooting. The RL controller adds power at high frequencies. This is the result of the waterbed effect, which implies that if we decrease the power at one part of the spectrum this always leads to the amplification of another part of the spectrum.

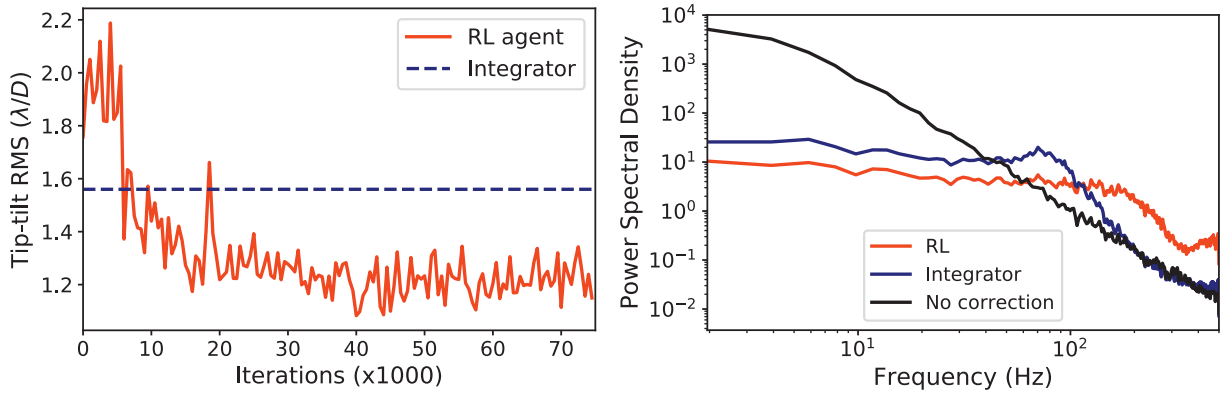

Fig 3 Left: Training curve of the Reinforcement Learning controller under power-law input turbulence and the average performance of an integrator with optimal gain. **Right:** Temporal PSD of the residuals in the x-direction for a simulation of 10 seconds.

4 Lab verification of tip-tilt control

4.1 Experimental Setup

The results in the previous section are based on idealized simulations. To validate our results for tip-tilt control we tested the algorithm in the lab. The lab setup is sketched in Figure 4. The first lens focuses a 633-nm He-Ne laser onto a single-mode fiber. The fiber output is then collimated, and a diaphragm defines the input aperture. A 4f system reimages the aperture onto a Holoeye LC2002 Spatial Light Modulator (SLM) with 800x600 pixels. The pupil covers approximately 200x200 pixels on the SLM. Right before the SLM we define the polarization with a polarizer. After a second polarizer we focus the beam onto the camera. The resulting Point Spread Function is shown in Figure 4.

We use the SLM as both the turbulence generator and the corrector. The polarizers are rotated such that the SLM mainly manipulates the phase profile with minimum amplitude modulation. The voltages of the SLM are controlled with 8-bit resolution. As the response for both small and large voltages becomes highly nonlinear, only values between 25 and 225 are used. Tip and tilt aberrations are introduced by applying a gradient on the SLM at the location of the pupil. Figure

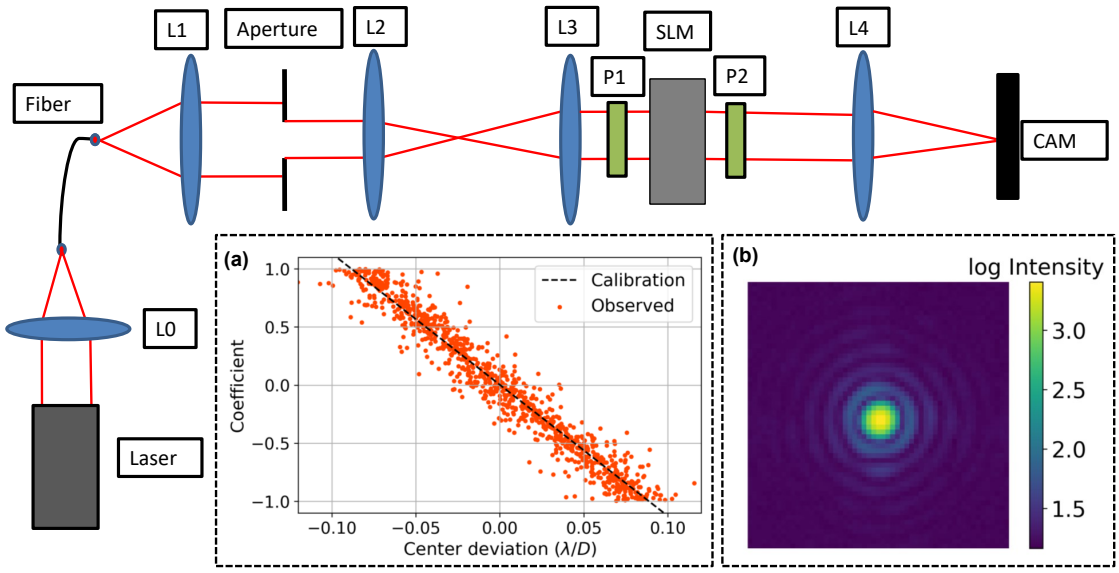

Fig 4 Lab setup to test the RL control algorithm with lenses L0-L4, Polarizers P1 and P2, a Spatial Light Modulator (SLM) and a camera (CAM). Insets: **(a)** Measured tilt in the focal plane as a function of the applied phase gradient on the SLM. **(b)** Measured Point Spread Function for the setup.

4 shows the calibration of the SLM for the resulting shift of the PSF in the focal plane image. The coefficient refers to the amplitude of the gradient profile, where 1 corresponds to the maximum possible gradient in the pupil. The maximum shift is approximately $\pm 0.1\ \lambda/D$. The refresh rate of the SLM is 30 Hz according to the manufacturer. We use the SLM at a frequency of approximately 18 Hz. We have attempted to measure the delay of the control loop but this appeared to not be constant. This might be the result of varying processing speeds in the pipeline. On average, there is a delay of 0.1 to 0.15 seconds, which is equivalent to 2-3 frames at 18 Hz. For the RL algorithm we use the same architectures, hyperparameters and reward function as in the simulations (see Section 3.1.1), except for the reward scaling b, for which we now use a value of $0.05\ \lambda/D$, because the shift in the PSF is smaller than in the simulations.

4.2 Vibration Suppression

We again test the ability of the algorithm to filter out a combination of 3 vibrations along each direction. We insert vibrations at 0.23, 0.48 and 0.9 Hz in the x-direction and at 0.18, 0.51 and 0.86 Hz in the y-direction. Figure 5 shows the training curve and the residual temporal PSD for the trained agent and an integrator with optimized gain. It also shows the residuals for a 500-iteration measurement. These results confirm our findings in the simulations as we again see a much reduced power in the vibrations. There is a little more power left in the vibrations as compared to the simulations, and the training time is longer. This is likely the result of the additional noise in the lab, variable control-loop delay and dynamics of the SLM.

4.3 Identification and mitigation of a varying vibration

So far, we have only considered disturbances with a stationary power spectrum. Once the input spectrum changes (e.g. the frequency of the vibrations), we would have to retrain the controller. However, Recurrent Neural Networks should be able to identify the relevant parameters online,

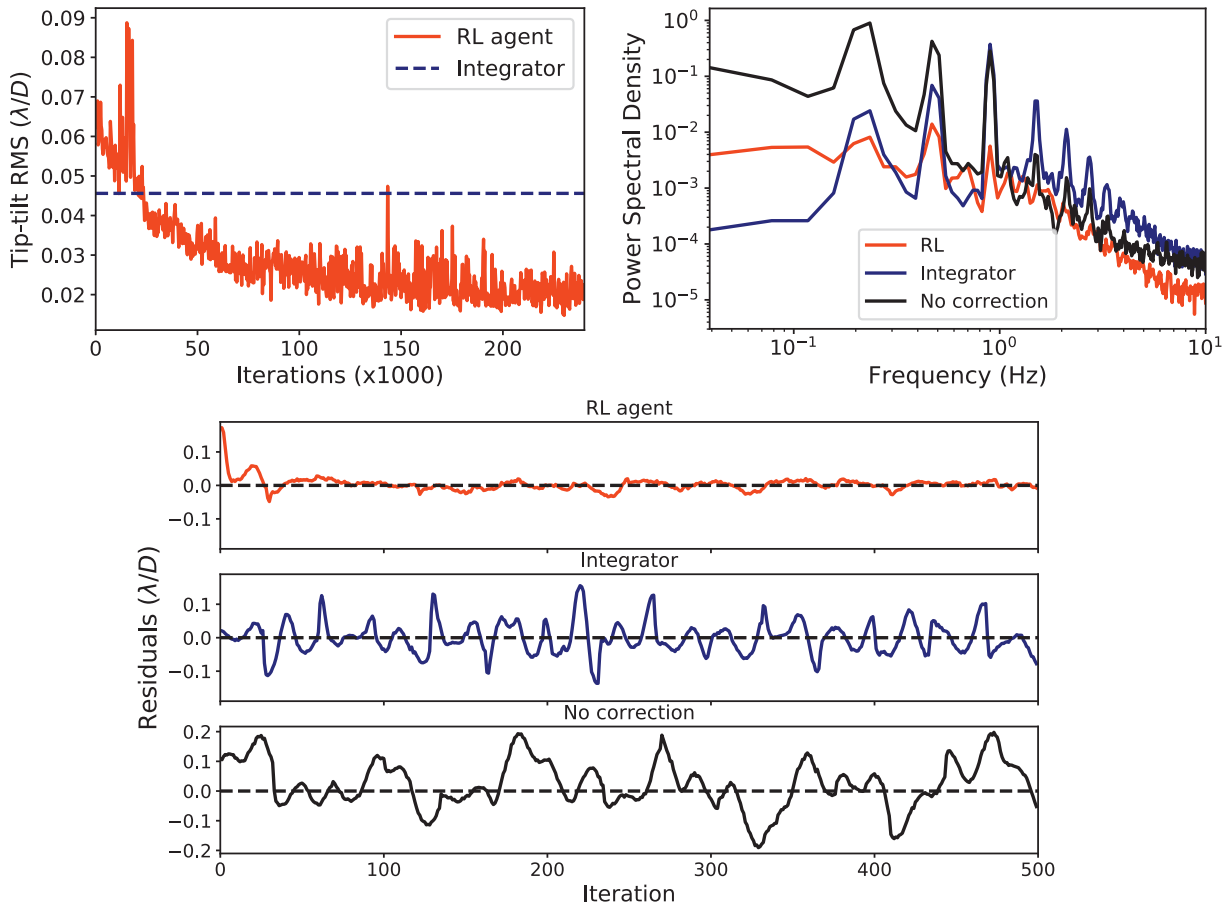

Fig 5 Top left: Training curve of the Reinforcement Learning controller for an input spectrum consisting of 3 vibrations along each direction in the lab. For comparison, we show the average performance of an integrator with optimal gain. **Top right:** Temporal PSD of the residuals in the x-direction for a measurement of 2500 iterations. **Bottom:** Residuals for the two controllers in the x-direction for a 500-iteration measurement.

without retraining. To test this, we train the controller with a single but randomly varying input vibration along the x-axis. After every 500 iterations we reset the vibration parameters: the amplitude is randomly sampled between 0 and 1 and the frequency between 0 and 1.8 Hz. The training curve is shown in Fig. 6. Once it has converged there is very little variance in the RMS, meaning that the performance is mostly independent of the amplitude and frequency of the input vibration. We compare the performance of the trained controller to that of an integrator with a fixed gain of 0.6. We apply a vibration with a fixed amplitude of 1 and a specific frequency and run closed loop for 2000 iterations. The residual RMS as a function of the frequency of the vibration is shown in Fig. 6. The data point for a frequency of 0 Hz is without an input vibration and is only due to natural turbulence on our optical table. The RL controller is able to reduce the RMS for all frequencies. On the other hand, the integrator amplifies the vibrations starting at a frequency of 1.25 Hz. This is the result of the vibration falling outside the correction bandwidth for this gain. However, the RNN is able to adaptively change its commands based on the observed sequence of measurements. This demonstrates that the RNN can learn to identify the relevant parameters of the disturbance, in this case the amplitude and frequency of the vibration, without updating the control law.

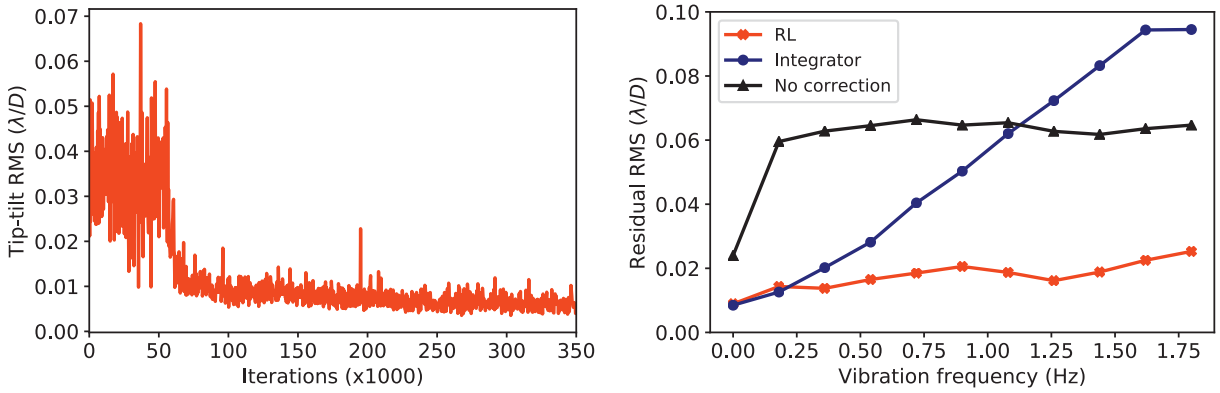

Fig 6 Left: Training curve of the Reinforcement Learning controller for a single vibration with random amplitude between 0 and 1 and random frequency between 0 and 1.8 Hz. **Right:** Residual RMS as a function of vibration frequency for a 2000-iteration measurement.

4.4 Power-law Disturbances

We also tested the performance for a power-law input spectrum. The training curve and residual temporal PSD are shown in Fig. 7. It takes many iterations before it converges. This is likely because of the noisy response of the SLM. We expect that with the improvements suggested in Section 5, this training time could be significantly reduced. Comparing the residual PSD to that of an integrator with optimized gain, we see results that are similar to the simulations, although the improvement is smaller. We again observe an increased rejection power at low frequencies without the bump due to overshooting.

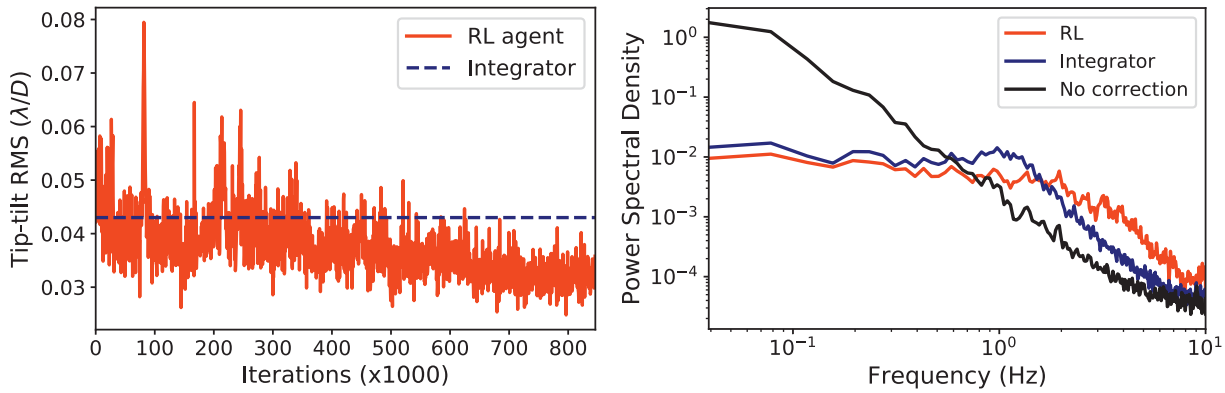

Fig 7 Left: Training curve of the Reinforcement Learning controller under power-law input turbulence in the lab along with the average performance of an integrator with optimal gain. **Right:** Temporal PSD of the residuals in the x-direction for a measurement of 2500 iterations.

5 Full wavefront control

5.1 Simulation setup

We simulate a typical high-contrast imaging instrument with an unobscured aperture with a diameter of 8 meters and a DM with 41x41 actuators operating at a loop frequency of 1 kHz. All simulations are again done using HCIPy.[44] We use a noiseless wavefront sensor, which directly senses the phase and project this onto the DM. Therefore, the control basis consists of Gaussian influence functions centered at the location of the DM actuators. This actuator basis has the advantage that the spatial response of each mode is the same but at a different location in the pupil. This results in each mode having the same spatio-temporal wavefront correlations, except for the actuators located at the edges. It also retains the spatial structure of the wavefront. The effective loop delay between sensing the wavefront and the application of the DM commands is 2 ms. To estimate the effect on the contrast, we propagate the residual wavefront through an ideal coronagraph.[50] The parameters of the simulations are shown in Table 4.

Table 4 Parameters of the AO simulations.

Parameter	Value
Aperture diameter	8 m
Wavelength	1 μm
Loop frequency	1 kHz
Servo-lag	2 ms
Number of illuminated actuators	1201

5.2 Architecture & reward

To take advantage of the spatio-temporal structure of the data we combine Recurrent Neural Networks and Convolutional Neural Networks (CNNs). CNNs are commonly used in image classification and regression problems[51,52] and assume that features are local and translationally invariant to drastically reduce the number of free parameters as compared to regular ANNs. In these CNNs,

kernels are convolved with the input data to obtain the activation in the next layer. In our case this means that we assume that each actuator has the same spatial response, and only close-by actuators are important for the prediction in the next step. We use a Convolutional Long Short-Term Memory (ConvLSTM) layer.[53] This is equivalent to a LSTM for each actuator with a local field of view and shared weights between the actuators. The recurrent part of the Convolutional LSTM is illustrated in Figure 8. In the case of a 1x1 kernel size, we would have the same decoupled control law for each individual actuator. However, this would not allow the algorithm to capture any spatial correlations in the data. Although these spatial correlations are found to be small in on-sky telemetry data for wind velocities $<$10 m/s on short timescales,[20] they may become more important for higher wind speeds or longer timescales. Furthermore, they may help in identifying the direction of the wind flow or to mitigate noise.

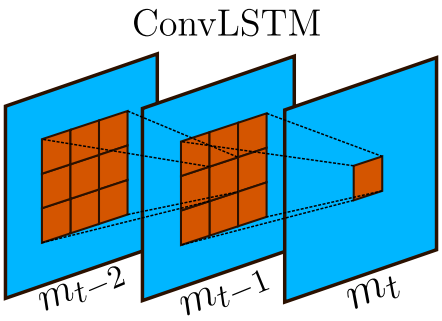

Fig 8 Illustration of the recurrent part of a Convolutional LSTM layer. The output at time t is determined by a local field of view of the memory states m_{t-1} and the local inputs.

As the reward, we would ideally want to directly optimize the Strehl ratio or the post-coronagraphic contrast in the science image. However, the accurate estimation of these quantities at kHz frequencies could be difficult. Therefore we use the squared residual wavefront error as the reward. This residual wavefront error is calculated from the WFS measurements. A disadvantage of this is that the reward suffers from the same limitations as the WFS. For example, it means that the algorithm is insensitive to modes to which the WFS is blind and suffers from the same aliasing effects as the WFS. Instead of using a single scalar as reward, we use a 2D reward map, with the squared residual error of each actuator mode at the actuator's location. This allows us to use a fully convolutional model for the critic, where only close-by actuators are considered when estimating the loss. This results in less free parameters and the access to more and localized information, leading to decreased training times as opposed to a global metric.

The input to our controller is the residual wavefront errors o_t and previous incremental DM commands a_t at the location of the actuator, concatenated along the depth, the number of inputs for each actuator. This means the input is an array of shape 41x41x2 for each time step. For unused actuators outside the pupil, we fill the array with zeros. Our actor architecture consists of a Convolutional LSTM layer with 16 3x3 kernels. After that, we have two regular convolutional layers with 8 and 1 3x3 kernels, respectively, which provide the new incremental DM commands a_t. For the critic, we again have a Convolutional LSTM layer with 16 3x3 kernels. After that we concatenate the DM commands to the output of the Convolutional LSTM. This is followed by three regular convolutional layers with 16, 8 and 1 3x3 kernels, respectively. The output of the critic is a 41x41x1 map of the expected return for each actuator. All convolutional layers use the tanh activation function except for the last layer of the Critic, which uses a linear activation function.

Table 5 Neural Network architectures of the actor and critic used for the full wavefront control.

	Layer type	Kernel size	Kernels	Input shape	Activation function
Actor	ConvLSTM	3x3	16	(41, 41, 2)	Tanh
	Conv	3x3	8	(41, 41, 16)	Tanh
	Conv	3x3	1	(41, 41, 8)	Tanh
	Layer type	Kernel size	Kernels	Input shape	Activation function
Critic	ConvLSTM	3x3	16	(41, 41, 2)	Tanh
	Concatenate	-	-	(41, 41,16), (41,41, 1)	-
	Conv	3x3	16	(41, 41, 17)	Tanh
	Conv	3x3	8	(41, 41, 16)	Tanh
	Conv	3x3	1	(41, 41, 8)	Linear

The architectures are summarized in Table 5. Both models have ∼12,000 free parameters. Again, we have not done an optimization of the architecture or hyperparameters.

5.2.1 Algorithm improvements

Instead of starting with a randomly initialized control law, as in our tip-tilt control, we pre-train the actor using supervised learning to mimic an integral controller with a sub-optimal gain of 0.4. This slightly reduces the training time. In addition, we add regularization to the DM commands to penalize the controller for potentially adding unsensed modes. We do this by adding an L_2 penalty term on the norm of the incremental DM commands to the loss function:

$$J(\theta) = \mathbb{E}[R] - \frac{1}{2}\lambda\mathbb{E}[a^2], \tag{16}$$

where λ is the strength of the regularization. The new policy gradient then becomes:

$$\nabla_\theta J(\theta) = \mathbb{E}[(\nabla_a Q_\omega(s,a) - \lambda a)\nabla_\theta \pi_\theta(s)]. \tag{17}$$

Finally, we slightly alter the training procedure. Before doing the truncated backpropagation through time, we first propagate the 10 previous states through the models. This ensures that the memory states are initialized correctly. Initialization was less of an issue for the tip-tilt control as the TBTT length was longer and therefore the initial timesteps contributed less to the total gradient as opposed to full AO control.

The hyperparameters for the algorithm used throughout this section are listed in Table 6. A notable change compared to the tip-tilt control is the lower discount factor γ because the optimization horizon does not need to be long. This is because of the monotonic nature of the reward in the absolute wavefront error; a command that decreases the wavefront error in the short term is also the best in the long-term. We still have to use $\gamma > 0$ because of the delay in the commands and reward. We also use a much higher target soft update τ and noise decay ζ, which both lead to decreased training times without giving up too much stability.

Table 6 Hyperparameters used for the full wavefront control

Parameter	Value
Actor learning rate	10^{-5}
Critic learning rate	10^{-3}
Target network soft update τ	0.1
Discount factor γ	0.9
Batch size	32
TBTT length l	20 ms
Action regularization λ	10^{-3}
Initial exploration σ_0	0.5 rad
Exploration decay ζ	0.1
Episode length	1000 iterations
Number of training steps per episode	1000

5.3 Stationary turbulence

First, we test the algorithm for input turbulence with stationary parameters. We simulate an atmosphere consisting of two layers of frozen-flow turbulence: A ground layer with a horizontal wind speed of 10 m/s with an r_0 of 15 cm at 500 nm and a jet stream layer with a wind speed of 30 m/s at an angle of 45 degrees with an r_0 of 20 cm at 500 nm. Here, an angle of 0 degrees corresponds to the turbulence flowing from left to right across the pupil. The average raw contrast behind the ideal coronagraph within the control radius during the training is shown in Figure 9. After training we test the algorithm without exploration noise. Figure 10 shows the resulting coronagraphic focal plane image for a 10-second simulation for our RL controller and an integrator with optimized gain. The gain of the integrator was optimized by running in closed loop for 3 seconds to determine the gain corresponding to the largest Strehl ratio. We find an optimal gain of 0.55 for the integrator. Figure 9 shows the radially averaged raw contrast profile for these images. The raw contrast is calculated as the average intensity in that radial bin divided by the Strehl ratio of the non-coronagraphic focal plane image. We observe an increase in raw contrast of up to two orders of magnitudes at small separations as compared to the integrator. Also shown is the contrast curve for an ideal controller, which perfectly removes all wavefront errors that can be fitted with the DM at each timestep, without any delay.

5.4 Non-stationary turbulence

One may wonder if the algorithm can learn to adapt to changing turbulence conditions on its own. To explore this, we train the algorithm under a variety of conditions. We randomly reset the wind speed and direction every second. The wind speed is uniformly sampled between 10 and 40 m/s, and we allow all angles for the wind direction. Only a single layer of frozen-flow turbulence is used in these simulations. The training curve is shown in Figure 11. After training, we test the algorithm for specific wind speeds and randomly sampled directions. The resulting focal plane images and contrast curves for a 10-second simulation are shown in Figure 12.

Our results indicate that the controller improves the contrast for all wind velocities and directions in these simulations. This demonstrates that the controller is able to learn to adapt to changes in the spatio-temporal PSD as a result of a different wind speed or angle. The controller therefore does not necessarily need online learning, in contrast to most other predictive control algorithms.

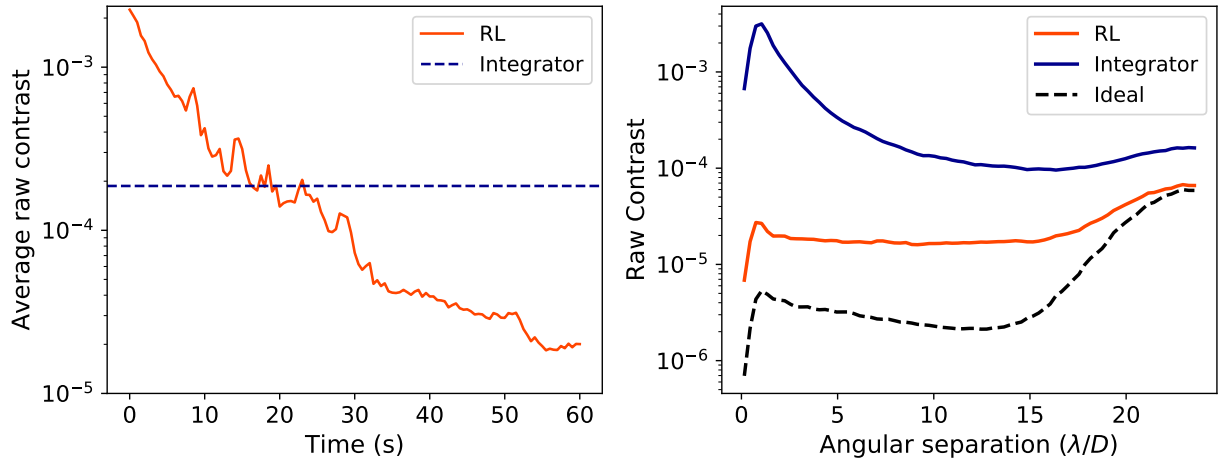

Fig 9 Left: Evolution of the average raw contrast after an ideal coronagraph within the control radius during the training of the RL controller for turbulence consisting of two layers of frozen-flow turbulence. **Right:** Raw contrast curves for a 10-second integration of the focal plane for the simulations with a two-layer atmosphere.

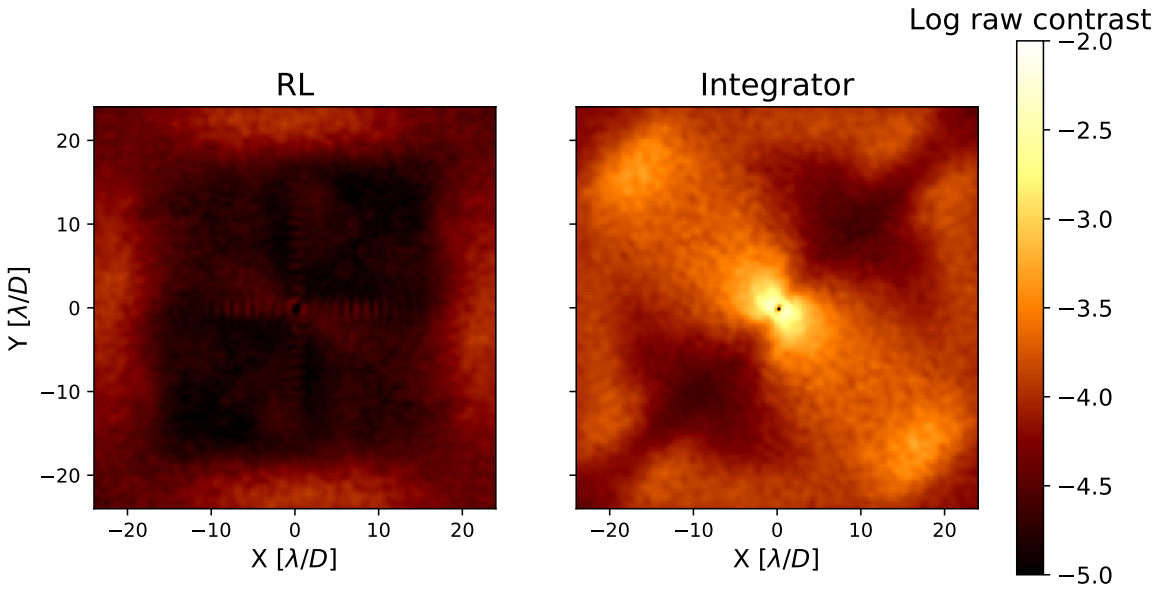

Fig 10 10-second integration of the focal plane after an ideal coronagraph for the simulations with an atmosphere consisting of two layers of frozen-flow turbulence.

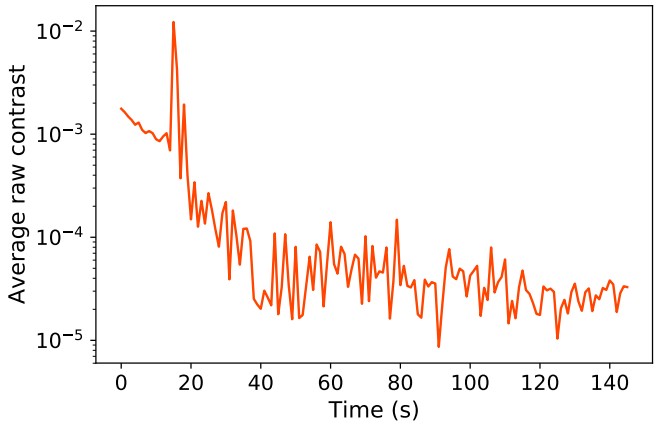

Fig 11 Evolution of the average raw contrast during the training phase behind an ideal coronagraph within the control radius for an atmosphere containing of a single layer of frozen-flow turbulence with random wind speed and direction.

However, the improvement in contrast is less than when optimized for a fixed wind speed and angle, as some wind-driven halo remains. This is because of the finite representation power of the model; a deeper or wider model will likely perform better, at the expense of added computational cost. Furthermore, this generalization would be even harder in the case of more complex turbulence with more varying parameters. An alternative approach may therefore be to start off with the general model and learn online for the current observing conditions. This may significantly reduce the time needed to train the controller. In the same way as adapting to different wind speeds, we also expect the algorithm to be able to adapt to different r_0 and account for the nonlinearities accordingly when using a Pyramid Wavefront Sensor. There is a residual cross-shaped ripple pattern in the coronagraphic images when using the RL controller. This is the result of square-edge effects, as the assumption that all actuators have the same spatial response does not hold for actuators at the edge. This can be seen in Fig. 13, which shows the residual wavefront RMS for a simulation with a wind speed of 30 m/s. One solution to this might be to not have all convolutional kernels have shared weights, at the cost of more free parameters.

6 Conclusions & Outlook

In conclusion, Reinforcement Learning is a promising approach towards data-driven predictive control for adaptive optics. We have shown how a closed-loop Recurrent Neural Network (RNN) controller can be trained using the Recurrent Deterministic Policy Gradient algorithm without any prior knowledge of the system dynamics and disturbances. This RNN only needs the most recent wavefront measurement and the previous DM commands at every time step.

First, we applied the algorithm to tip-tilt control for a simulated, ideal AO system. We demonstrated that our approach can learn to mitigate a combination of tip-tilt vibrations, reducing the residual RMS by a factor of ~ 6 as compared to an optimal-gain integrator. We also show a $\sim 25\%$ decrease in residual RMS for power-law input turbulence compared to the optimal gain integrator. These experiments were then repeated in a lab setup. For the vibration mitigation we observe a decrease in RMS by a factor 2.2 and a $\sim 25\%$ decrease in residual RMS for power-law input turbulence. Furthermore, we showed that a single Recurrent Neural Network controller can

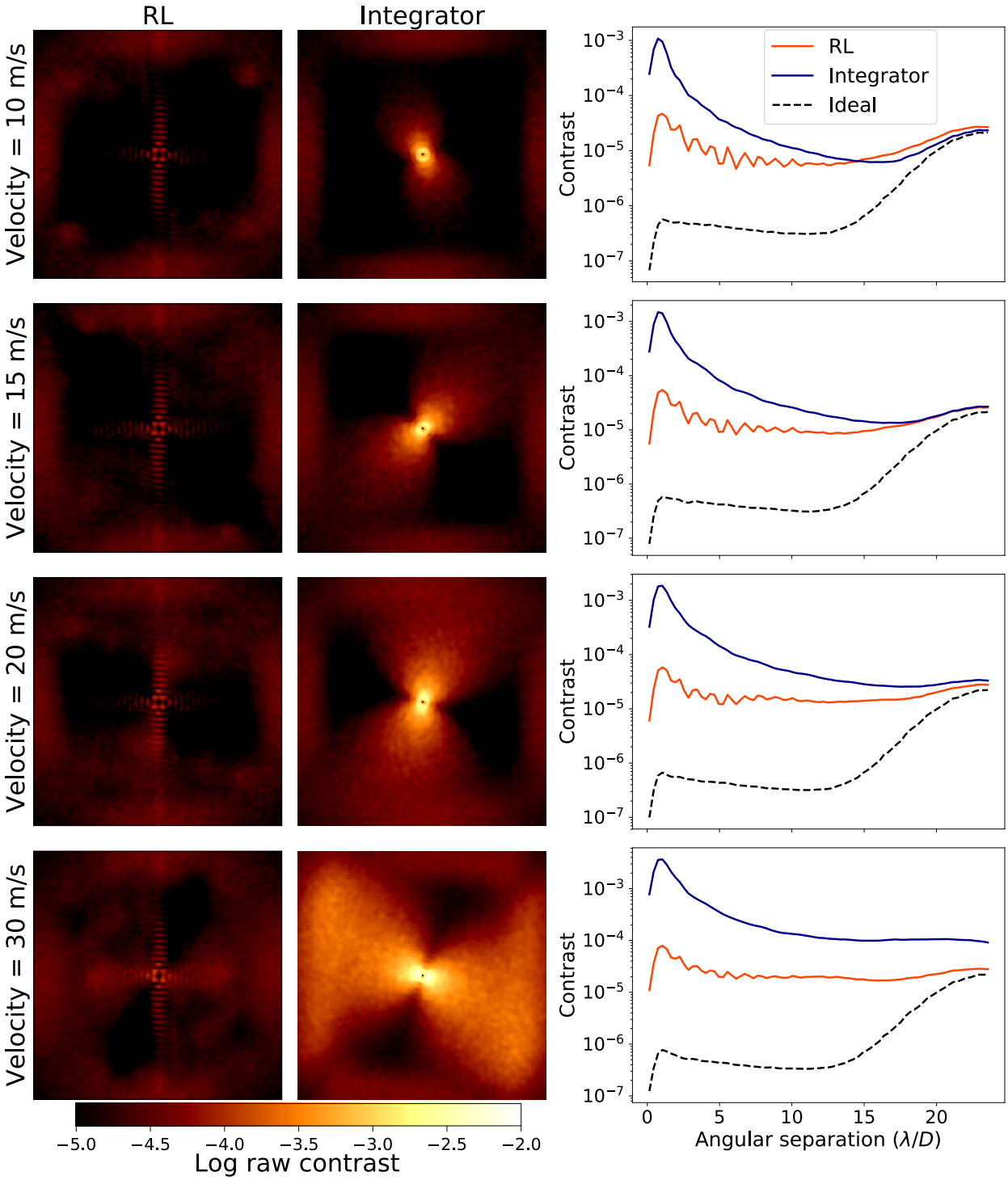

Fig 12 Resulting coronagraphic focal-plane images and contrast curves for a 10 second simulation for a specific wind velocity and a random wind direction. Each row represents a given wind velocity, which is listed on the left. The left column shows the image using the Reinforcement Learning controller, the middle column the image obtained while using an integrator with optimized gain and the right column shows the resulting contrast curves.

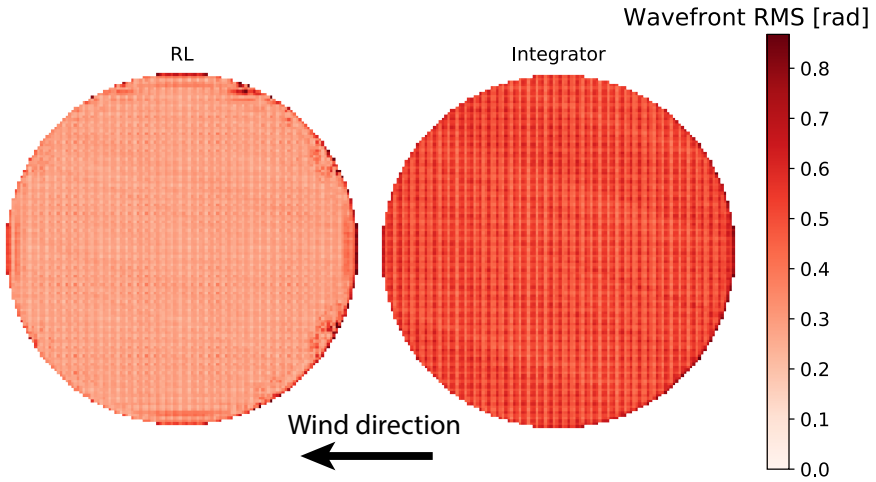

Fig 13 Residual wavefront RMS for a wind speed of 30 m/s with the controller trained on random wind directions.

mitigate a vibration with varying amplitude and frequency without needing online updating of the control parameters.

Secondly, we showed in simulations how the algorithm can be applied to the control of a high-order DM. We showed that for an atmosphere consisting of a ground layer and a jet-stream layer, the algorithm can improve the contrast at small separations by two orders of magnitude as compared to an optimal-gain integrator. Furthermore, when the controller is trained over a variety of observing conditions, it is able to adapt to changes in the wind speed and direction without needing online updating of the control law. This relaxes the real-time computational demand that would be required to update the control law given the constantly changing atmospheric conditions. It also simplifies the implementation because control and learning can be decoupled. However, the obtained contrast is lower than when trained for stationary conditions. A deeper or wider model may improve its ability to generalize and perform under varying conditions, at the expense of added computational time.

We have not considered the effect of photon noise on the performance of the algorithm. As the noise affects both the state and the reward, we expect that this will increase the amount of data needed to train the controller. Future research should test the effect of noise on the performance of the controller. The algorithm should also be tested with a nonlinear wavefront sensor, such as the pyramid wavefront sensor, because our approach has the advantage of easily accommodating a nonlinear control law. Furthermore, the algorithm is off-policy, meaning that we can train on data that is not collected with the current optimal control law. We can therefore use historical telemetry data collected using an integrator to train the controller. It should be investigated if this can already provide a good controller or if we need to learn online.

Although we have shown the potential of Reinforcement Learning for closed-loop predictive control, there are a few more practical considerations. First is the complexity of the implementation of the algorithm on real-time controllers (RTC) at actual telecopes, because current RTCs are often not compatible with the Deep Learning interfaces used in this paper. Another consideration is the computational cost. Although prediction is easily within the capabilities of current RTCs because of the partially decoupled approach, online learning with backpropagation will be computationally very challenging. For example, the full training process, including optical simulations and gradient updates, for the full wavefront control in the stationary case (see Section 5.3) took

about 4 hours using the NVIDIA Tesla K80 GPU offered by Google Colab. While this can be sped up significantly by the use of multiple GPU's or specialized hardware, training times are likely to increase under more complex turbulence and with real noise. However, we have shown that the algorithm can learn to perform under varying conditions, allowing control and training to be decoupled, making the computational cost of training less of an issue. Alternatively, one could use the generalized model and finetune it for the current observing conditions, significantly reducing the training time. Another consideration is the stability of the learning algorithm and controller. Since we are using a sophisticated nonlinear controller, it is difficult to give performance guarantees. Furthermore, changing the hyperparameters can significantly influence the learning stability of the algorithm.

Acknowledgements

We wish to thank the reviewers for their feedback, which has resulted in improvements of this work. R.L. acknowledges funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under grant agreement No 694513. Support for this work was provided by NASA through the NASA Hubble Fellowship grant #HST-HF2-51436.001-A awarded by the Space Telescope Science Institute, which is operated by the Association of Universities for Research in Astronomy, Incorporated, under NASA contract NAS5-26555. Part of this work was already presented in Ref. 54.

Disclosures

The authors declare no conflict of interest.

References

1 O. Guyon, "Extreme adaptive optics," *Annual Review of Astronomy and Astrophysics* **56**(1), 315–355 (2018).

2 F. Cantalloube, O. J. Farley, J. Milli, *et al.*, "Wind-driven halo in high-contrast images: I. Analysis of the focal-plane images of SPHERE," *Astronomy and Astrophysics* **638** (2020).

3 J. Lozi, O. Guyon, N. Jovanovic, *et al.*, "Characterizing vibrations at the Subaru Telescope for the Subaru coronagraphic extreme adaptive optics instrument," *Journal of Astronomical Telescopes, Instruments, and Systems* **4**(4), 1–13 (2018).

4 M. Hartung, T. Hayward, L. Saddlemyer, *et al.*, "On-sky vibration environment for the Gemini Planet Imager and mitigation effort," in *Adaptive Optics Systems IV*, E. Marchetti, L. M. Close, and J.-P. Véran, Eds., **9148**, 202 – 213, International Society for Optics and Photonics, SPIE (2014).

5 J.-F. Sauvage, T. Fusco, C. Petit, *et al.*, "SAXO, the eXtreme Adaptive Optics System of SPHERE: overview and calibration procedure," in *Adaptive Optics Systems II*, B. L. Ellerbroek, M. Hart, N. Hubin, *et al.*, Eds., **7736**, 175 – 184, International Society for Optics and Photonics, SPIE (2010).

6 J. P. Lloyd and A. Sivaramakrishnan, "Tip-Tilt Error in Lyot Coronagraphs," *The Astrophysical Journal* **621**, 1153–1158 (2005).

7 D. Mawet, L. Pueyo, D. Moody, *et al.*, "The Vector Vortex Coronagraph: sensitivity to central obscuration, low-order aberrations, chromaticism, and polarization," in *Modern Technologies in Space- and Ground-based Telescopes and Instrumentation*, E. Atad-Ettedgui and D. Lemke, Eds., **7739**, 378–390, International Society for Optics and Photonics, SPIE (2010).

8 E. Gendron and P. Lena, "Astronomical adaptive optics. I. Modal control optimization," *Astronomy and Astrophysics* **291**(1), 337–337 (1994).

9 E. Gendron and P. Lena, "Astronomical adaptive optics. II. Experimental results of an optimized modal control.," *Astronomy and Astrophysics Supplement Series* **111**(111), 153 (1995).

10 R. N. Paschall and D. J. Anderson, "Linear quadratic Gaussian control of a deformable mirror adaptive optics system with time-delayed measurements," *Applied Optics* **32**(31), 6347 (1993).

11 C. Correia, J.-P. Véran, and G. Herriot, "Advanced vibration suppression algorithms in adaptive optics systems," *Journal of the Optical Society of America A* **29**(3), 185 (2012).

12 C. Petit, J.-M. Conan, C. Kulcsár, *et al.*, "First laboratory validation of vibration filtering with LQG control law for Adaptive Optics," *Optics Express* **16**(1), 87 (2008).

13 G. Sivo, C. Kulcsár, J.-M. Conan, *et al.*, "First on-sky SCAO validation of full LQG control with vibration mitigation on the CANARY pathfinder," *Optics Express* **22**(19), 23565 (2014).

14 C. Petit, J.-F. Sauvage, T. Fusco, *et al.*, "SPHERE eXtreme AO control scheme: final performance assessment and on sky validation of the first auto-tuned LQG based operational system," in *Adaptive Optics Systems IV*, E. Marchetti, L. M. Close, and J.-P. Véran, Eds., **9148**, 214 – 230, International Society for Optics and Photonics, SPIE (2014).

15 C. Correia, H.-F. Raynaud, C. Kulcsár, *et al.*, "On the optimal reconstruction and control of adaptive optical systems with mirror dynamics," *Journal of the Optical Society of America A* **27**(2), 333 (2010).

16 L. A. Poyneer, D. W. Palmer, B. Macintosh, *et al.*, "Performance of the Gemini Planet Imager's adaptive optics system," *Applied Optics* **55**(2), 323 (2016).

17 C. Dessenne, P.-Y. Madec, and G. Rousset, "Modal prediction for closed-loop adaptive optics," *Optics Letters* **22**(20), 1535 (1997).

18 O. Guyon and J. Males, "Adaptive Optics Predictive Control with Empirical Orthogonal Functions (EOFs)," (2017).

19 R. Jensen-Clem, C. Z. Bond, S. Cetre, *et al.*, "Demonstrating predictive wavefront control with the Keck II near-infrared pyramid wavefront sensor," in *Techniques and Instrumentation for Detection of Exoplanets IX*, S. B. Shaklan, Ed., **11117**, 275 – 284, International Society for Optics and Photonics, SPIE (2019).

20 M. A. Van Kooten, N. Doelman, and M. Kenworthy, "Robustness of prediction for extreme adaptive optics systems under various observing conditions: An analysis using VLT/SPHERE adaptive optics data," *Astronomy and Astrophysics* **636**, 81 (2020).

21 V. Deo, É. Gendron, G. Rousset, *et al.*, "A telescope-ready approach for modal compensation of pyramid wavefront sensor optical gain," *Astronomy & Astrophysics* **629**, A107 (2019).

22 A. P. Wong, B. R. M. Norris, P. G. Tuthill, *et al.*, "Predictive control for adaptive optics using neural networks," *Journal of Astronomical Telescopes, Instruments, and Systems* **7**(1), 1–22 (2021).

23 R. Swanson, M. Lamb, C. M. Correia, *et al.*, "Closed Loop Predictive Control of Adaptive Optics Systems with Convolutional Neural Networks," *Monthly Notices of the Royal Astronomical Society* (2021).

24 S. Y. Haffert, J. R. Males, L. M. Close, *et al.*, "Data-driven subspace predictive control of adaptive optics for high-contrast imaging," *Journal of Astronomical Telescopes, Instruments, and Systems* **7**(2), 1 – 22 (2021).

25 M. van Kooten, N. Doelman, and M. Kenworthy, "Impact of time-variant turbulence behavior on prediction for adaptive optics systems," *J. Opt. Soc. Am. A* **36**, 731–740 (2019).

26 M. Gray, C. Petit, S. Rodionov, *et al.*, "Local ensemble transform Kalman filter, a fast non-stationary control law for adaptive optics on ELTs: theoretical aspects and first simulation results," *Optics Express* **22**(17), 20894 (2014).

27 X. Liu, T. Morris, C. Saunter, *et al.*, "Wavefront prediction using artificial neural networks for open-loop adaptive optics," *Monthly Notices of the Royal Astronomical Society* **496**(1), 456–464 (2020).

28 R. Landman and S. Y. Haffert, "Nonlinear wavefront reconstruction with convolutional neural networks for Fourier-based wavefront sensors," *Optics Express* **28**(11), 16644 (2020).

29 B. R. M. Norris, J. Wei, C. H. Betters, *et al.*, "An all-photonic focal-plane wavefront sensor," *Nature Communications* (2020), 1–9 (2020).

30 R. Swanson, M. Lamb, C. Correia, *et al.*, "Wavefront reconstruction and prediction with convolutional neural networks," in *Adaptive Optics Systems VI*, L. M. Close, L. Schreiber, and D. Schmidt, Eds., **10703**, 481 – 490, International Society for Optics and Photonics, SPIE (2018).

31 V. M. Radhakrishnan, C. U. Keller, and N. Doelman, "Optimization of contrast in adaptive optics for exoplanet imaging," in *Adaptive Optics Systems VI*, L. M. Close, L. Schreiber, and D. Schmidt, Eds., **10703**, 1211 – 1217, International Society for Optics and Photonics, SPIE (2018).

32 H. Sun, N. J. Kasdin, and R. Vanderbei, "Identification and adaptive control of a high-contrast focal plane wavefront correction system," *Journal of Astronomical Telescopes, Instruments, and Systems* **4**(04), 1 (2018).

33 J. Nousiainen, C. Rajani, M. Kasper, *et al.*, "Adaptive optics control using model-based reinforcement learning," *Optics Express* **29**, 15327 (2021).

34 R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press (2018).

35 K. Arulkumaran, M. P. Deisenroth, M. Brundage, *et al.*, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine* **34**(6), 26–38 (2017).

36 D. Silver, G. Lever, N. Heess, *et al.*, "Deterministic policy gradient algorithms," *31st International Conference on Machine Learning, ICML 2014* **1**, 605–619 (2014).

37 T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *4th International Conference on Learning Representations, ICLR 2016* (2016).

38 I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, MA, USA (2016). http://www.deeplearningbook.org.

39 R. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society* **60**(6), 503 – 515 (1954).

40 D. Wierstra, A. Foerster, J. Peters, *et al.*, "Solving deep memory pomdps with recurrent policy gradients," in *Proceedings of the 17th International Conference on Artificial Neural Networks*, *ICANN'07*, 697–706, Springer-Verlag, (Berlin, Heidelberg) (2007).
41 N. Heess, J. J. Hunt, T. P. Lillicrap, *et al.*, "Memory-based control with recurrent neural networks," (2015).
42 A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 6645–6649 (2013).
43 S. Hochreiter and J. Urgen Schmidhuber, "Long Short-term Memory," *Neural Computation* **9**(8), 17351780 (1997).
44 E. H. Por, S. Y. Haffert, V. M. Radhakrishnan, *et al.*, "High Contrast Imaging for Python (HCIPy): an open-source adaptive optics and coronagraph simulator," in *Adaptive Optics Systems VI*, *Proc. SPIE* **10703**, 152 (2018).
45 D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations, ICLR 2015* (2015).
46 M. Abadi, A. Agarwal, P. Barham, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," (2015). Software available from tensorflow.org.
47 F. Chollet *et al.*, "Keras." https://keras.io (2015).
48 P. D. Welch, "The Use of Fast Fourier Transform for the Estimation of Power Spectra," *Digital Signal Processing* (2), 532–574 (1975).
49 J.-M. Conan, G. Rousset, and P.-Y. Madec, "Wave-front temporal spectra in high-resolution imaging through turbulence," *J. Opt. Soc. Am. A* **12**, 1559–1570 (1995).
50 O. Guyon, E. A. Pluzhnik, M. J. Kuchner, *et al.*, "Theoretical Limits on Extrasolar Terrestrial Planet Detection with Coronagraphs," *The Astrophysical Journal Supplement Series* **167**(1), 81–99 (2006).
51 A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems* **25**, 1097–1105 (2012).
52 K. He, X. Zhang, S. Ren, *et al.*, "Deep residual learning for image recognition," (2015).
53 X. Shi, Z. Chen, H. Wang, *et al.*, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in Neural Information Processing Systems* **28**, 802–810 (2015).
54 R. Landman, S. Y. Haffert, V. M. Radhakrishnan, *et al.*, "Self-optimizing adaptive optics control with reinforcement learning," in *Adaptive Optics Systems VII*, L. Schreiber, D. Schmidt, and E. Vernet, Eds., **11448**, 842 – 856, International Society for Optics and Photonics, SPIE (2020).

Rico Landman is a PhD candidate at Leiden Observatory working on direct imaging and spectroscopy of exoplanets. He received his BS degrees in physics and astronomy and MS degree in astronomy from Leiden University in 2018 and 2020, respectively.

Sebastiaan Y. Haffert is a NASA Hubble Postdoctoral Fellow at the University of Arizona's Steward Observatory. His research focuses on high-spatial and high-spectral resolution instrumentation for exoplanet characterization.

Vikram M. Radhakrishnan is a PhD candidate at Leiden Observatory working on innovative control approaches to high-contrast imaging.

Christoph U. Keller is a Professor of Experimental Astrophysics at Leiden Observatory. He specializes in developing innovative optical instruments for astronomy, remote sensing and biomedical imaging.

List of Figures

1 Visual overview of the algorithm. The black line indicates forward propagation. The green and red lines indicate how we backpropagate the gradients to update the critic and the actor, respectively.

2 **Top left:** Training curve of the Reinforcement Learning controller under input disturbance consisting of three vibrations along both directions. Also shown is the average performance of an integrator with optimal gain. **Top right:** Temporal PSD of the residuals in the x-direction for a simulation of 10 seconds. Note that the lines of the input spectrum and integrator overlap at the peak at 37 Hz. **Bottom:** Residuals for the different controllers in the x-direction for a 500 ms simulation.

3 **Left:** Training curve of the Reinforcement Learning controller under power-law input turbulence and the average performance of an integrator with optimal gain. **Right:** Temporal PSD of the residuals in the x-direction for a simulation of 10 seconds.

4 Lab setup to test the RL control algorithm with lenses L0-L4, Polarizers P1 and P2, a Spatial Light Modulator (SLM) and a camera (CAM). Insets: **(a)** Measured tilt in the focal plane as a function of the applied phase gradient on the SLM. **(b)** Measured Point Spread Function for the setup.

5 **Top left:** Training curve of the Reinforcement Learning controller for an input spectrum consisting of 3 vibrations along each direction in the lab. For comparison, we show the average performance of an integrator with optimal gain. **Top right:** Temporal PSD of the residuals in the x-direction for a measurement of 2500 iterations. **Bottom:** Residuals for the two controllers in the x-direction for a 500-iteration measurement.

6 **Left:** Training curve of the Reinforcement Learning controller for a single vibration with random amplitude between 0 and 1 and random frequency between 0 and 1.8 Hz. **Right:** Residual RMS as a function of vibration frequency for a 2000-iteration measurement.

7 **Left:** Training curve of the Reinforcement Learning controller under power-law input turbulence in the lab along with the average performance of an integrator with optimal gain. **Right:** Temporal PSD of the residuals in the x-direction for a measurement of 2500 iterations.

8 Illustration of the recurrent part of a Convolutional LSTM layer. The output at time t is determined by a local field of view of the memory states m_{t-1} and the local inputs.

9 **Left:** Evolution of the average raw contrast after an ideal coronagraph within the control radius during the training of the RL controller for turbulence consisting of two layers of frozen-flow turbulence. **Right:** Raw contrast curves for a 10-second integration of the focal plane for the simulations with a two-layer atmosphere.
10 10-second integration of the focal plane after an ideal coronagraph for the simulations with an atmosphere consisting of two layers of frozen-flow turbulence.
11 Evolution of the average raw contrast during the training phase behind an ideal coronagraph within the control radius for an atmosphere containing of a single layer of frozen-flow turbulence with random wind speed and direction.
12 Resulting coronagraphic focal-plane images and contrast curves for a 10 second simulation for a specific wind velocity and a random wind direction. Each row represents a given wind velocity, which is listed on the left. The left column shows the image using the Reinforcement Learning controller, the middle column the image obtained while using an integrator with optimized gain and the right column shows the resulting contrast curves.
13 Residual wavefront RMS for a wind speed of 30 m/s with the controller trained on random wind directions.

List of Tables

1 Overview and explanation of the various terms used in the Reinforcement Learning (RL) framework.
2 Neural Network architectures of the actor and critic for the tip-tilt control.
3 Hyperparameters
4 Parameters of the AO simulations.
5 Neural Network architectures of the actor and critic used for the full wavefront control.
6 Hyperparameters used for the full wavefront control