Deep reinforcement learning for tiled aperture beam combining in a simulated environment

Henrik Tünnermann and Akira Shirakawa
Institute for Laser Science (ILS), University of Electro-Communications (UEC), 1-5-1 Chofugasoka, Chofu-shi, Tokyo 182-8585, Japan
E-mail: tuennermann@gmail.com

Abstract
Coherent beam combining is a method for combining multiple emitters into one high power beam by means of relative phase stabilization. Usually, modulation or interferometric techniques are used to generate an error signal. This is relatively complicated and expensive. Especially in the case of tiled aperture combining the beam profile is usually monitored anyway. This beam profile should contain most of the information necessary for the stabilization as well but is usually not used because it is difficult to explicitly derive the correct actions from just the far-field image. Here we show that it is possible to derive a suitable control policy without any explicit modeling using deep reinforcement learning in a simulated environment.

1. Introduction
Coherent beam combining (CBC) is a method for combining multiple emitters into one high power beam by means of relative phase stabilization. Usually, modulation or interferometric techniques are used to generate an error signal [1–3]. There are three arguments that can be made in favor of machine-learning based feedback methods for CBC and many other control tasks in optics:

First, it could be more simple to set up than a classic control scheme while still performing adequately well. Since a machine-learning based technology will arguably never be more simple than a proportional integral derivative controller therefore it will only be useful for more complicated cases.

Secondly, a machine-learning based approach can model the system as it is observed by the data and make predictions about it. So instead of just reacting to input as fast as possible, model-predictive control without manually specifying a model is feasible. We explored this in our last publication [4], in which we have shown that reinforcement learning (RL) is suitable for CBC and that predictive control is feasible, but will only have a significant impact on very slow feedback loops.

Thirdly, there is the error signal generation. This tends to be the most complicated part of CBC control. Classically modulation methods such as LOCSET [5] or the Hänsch-Culliard [6] method are used. Both work well but have limitations especially for pulsed laser systems. Here, machine learning methods could present an advantage due to the fact that any data related to the output can be used. In this manuscript we will discuss the stabilization of tiled aperture beam combining, however stabilizing photonic lanterns or beam quality could also be feasible with a similar design.

The second and third point can potentially lead to control systems with are not feasible by any other means. We are going to explore the third point in this manuscript in the context of tiled aperture beam combining.

The manuscript is organized as follows: in the next section, we will briefly review RL with a focus on optics. While there are many good resources on the topic [7, 8] this section will explain the basics necessary to understand the rest of the manuscript. After this, we will describe tiled aperture beam combining, and the feedback path and the results before discussing further options enabled by this method such as beam shaping.
2. Deep reinforcement learning

Reinforcement learning tries to learn the optimal action $a_t$ (in case of CBC this is the actuator movement) given a state $s_t$ at time $t$, in order to maximize a total future reward $r$. The state $s_t$ is either directly an observation or modified by an observation. Each such pair of actions and states one can then be assigned a value $Q(s_t, a_t)$. The better the action $a_t$ in state $s_t$ the higher the $Q$. This means if we know $Q$ we can determine the best possible action $a_t^{opt}$, because $Q(s_t, a_t^{opt}) = \max_a Q(s_t, a_t)$. The problem is we do not know $Q$ of course. We can approximate $Q$ by starting with a random $Q$ and iteratively making it better by applying

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a)),$$

Here, $\alpha$ is the learning rate—which determines how fast we change $Q$ at each step and $\gamma$ is the discount factor which, all other things being equal, accounts for the fact that we care more about rewards now than in the future. It has to be less than 1 for stability. We can represent $Q$ as a neural network. In this case, we can simply turn the problem into the usual supervised learning form with observations as $x$ and targets as $y$. Those are easy to calculate if we have $s_t$, $s_{t+1}$, $r_t$ and the neural network $Q$ calculates a vector of values representing the potential actions:

$$x = [s_t, a_t]$$
$$y = r_t + \gamma \max_a Q_{old}(s_{t+1}, a).$$

In practice there are two potential problems here: Neural networks assume uncorrelated samples when training. This will never be true for RL as samples collected at similar times will always be highly correlated. To avoid this we need to save the $s_t, s_{t+1}, r_t$ values, and sample from the old data pool to get uncorrelated observations. There are more advanced methods that prefer to use samples with high error but simply taking random state action pairs from the buffer is enough.

The second potential problem is that we use $Q$ to calculate $y$ which we then use to update $Q$, which can become unstable. To stabilize this is can be a good idea to use two $Q$ networks.

At this point, we have deep RL for discrete action spaces. However, most action spaces in the physical world are continuous which means we will need a method to account for this.

The most simple way to get a continuous actions space is by discretizing it and mapping the analog values onto a vector of finite numbers. We used this method in our first experiments. However, there are better options, most notably the deep deterministic policy gradient (DDPG) \cite{DDPG}. The general architecture of this scheme is shown in figure 1.
3. Tiled aperture beam combining

There are two main architectures of CBC: tiled- and filled-aperture. Filled-aperture uses beamsplitters, polarizers, or gratings to combine the beams in the near and far-field. This results in diffraction-limited beam quality provided that the single emitters are diffraction-limited. On the other hand, it is also possible to simply pack the individual emitters as tightly as possible and propagate the image to the far-field. If all of the emitters are in phase this will result in a strong center intensity and several minor side lobes. Obviously, this will never result in diffraction-limited beam quality, but it enables beam shaping and steering via control of the optical phase and is overall very simple and scalable. To realize it one simply needs a fiber bundle and a lens array with high fill factor or a multicore fiber as shown in figure 2.

Tiled aperture beam combining is a particularly interesting application of RL as it has some quite unique properties. Usually, we want to stabilize the far-field beam profile. The classic way to achieve this is either by applying LOCSET or the use of a camera-based wavefront sensing [11]. Both methods aim to reconstruct the phase of the electric field while what is really wanted is just a stable far field.

This is because it is not obvious what action to take to get closer to the desired state from the far-field beam profile. If only the intensity of far-field is known, there is no one to one mapping of near field phase patterns. In most configurations, it is very simple to find where different phase patterns yield the same far-field intensity pattern (for one example see the donut-shaped beam in section 5). This means we cannot always know the phase of all the emitters given a far-field pattern and more specifically that simply running a phase reconstruction algorithm (besides being too slow and not adapting to noise well) will not work while RL should be able to help us observe and memorize beneficial actions given a far-field intensity pattern anyway. Because the far field does not contain the full phase information of the near field, the history of actuator feedback and corresponding pattern changes could be necessary, however we found that this is not the case and we can simply use a single image.

We tested this in a simulated environment by propagating a near field pattern into the far-field. This can be easily done by the split-step Fourier method. We used a 64×64 image of the far-field as the representation of the state and our action is characterized by the change of the actuator induced phase shift in radian. For reliability and comparability we relied on the DDPG implementation of rllib [12]. As the reward function we used the negative phase difference between the emitters

$$R = -\sum_{i=1}^{n-1} (\phi_i - \phi_0)^2.$$  \hspace{1cm} (3)

While this is not the most practical reward function for experiments, it allows to easily compare different emitter configurations and number of emitters, which will provide valuable insight into the capabilities and limitations of this method. The reward function (3) becomes 0 if all emitters are in phase. The worst case for n emitters is $r_{\text{min}} = -(n-1) \cdot \pi^2$ and for random phase the mean reward is $\langle r \rangle = -(n-1) \cdot \pi^2/3$. These expressions scale with $n-1$ because there are $n-1$ relative phases. This becomes clear when looking at one core which is always in phase with itself. Therefore, when comparing convergence, one should normalize by the number of relative phases by dividing by $(n-1)$.

The simulated light is 1064 nm and the individual cores were 20 µm diameter and 50 µm pitch. We tried other configurations as shown in figure 3.

We can see from figure 4, which shows the training of the DDPG algorithm for three to seven emitters. The same algorithm was able to work with different configurations and even with a different number of emitters. We can also see that the amount of training samples seems to increase (with the exception of the six core configuration) with the number of emitters that have to be controlled. This is a well-known property of RL and one that makes scaling to large action spaces challenging. However, in our example we can also see
that only about 60 000 training samples are needed, which is a surprisingly low number which can be acquired in an experiment very quickly. So unlike many robotics tasks, here the data acquisition rate from the experiment does not provide a major challenge. However, when we tried to increase the emitter count to 19 emitters we noticed that dimensionality begins to be challenging. We estimate that for scaling to more than 100 emitters advanced methods like pre-training or expert guidance will be needed. We also have to note that the DDPG algorithm is comparatively data efficient as it is a so-called off policy method which allows us to re-use old data. Of course, this only tells us that the neural network learns to converge to a state that allows it to optimize the phase efficiently. To see how efficiently we saved it and tested is under varying noise levels.

In a simulation the image does not move so we can simply use a three dense layer neural network. In a more realistic environment maybe convolutional layers could make the trained algorithm more robust against drifts, which result in a movement of the image but this was not necessary in this case. As we can see only using a single image together with three dense layers works surprisingly well (figure 5), especially considering that the phase cannot be completely reconstructed. As we can see from figure 4 the neural network architecture furthermore works regardless of the actual configuration and target.

The next question to answer is how well does the neural network algorithm compensate the noise. Here two properties are important: how fast we can run a loop and how much we can compensate in one such step. How fast we can run a loop depends on the actual experimental hardware and there mostly on the speed of the actuator and analog-digital conversion. The time of the neural network calculation was designed to be negligible in comparison. Since we are running in a simulated environment, the best way to evaluate would be to simulate the noise as Gaussian random walk noise with different variance $\sigma$. This way on the one hand

Figure 3. Input near field data. Each spot represents an emitter with an individual phase.

Figure 4. Convergence of the episode reward (one episode is 200 steps and normalized by emitter count. The expected reward for one episode at the beginning of training is therefore $-200 \cdot \pi^2/3$).
we get a good approximation or real-world phase noise and are on the other hand able to scale the noise per time-step easily. The mean phase difference for the seven core is shown in figure 5. The control is turned on at time-step 1000. Typically it should be possible to run the correction sufficiently often so the noise of a single timestep does not exceed $\sigma = 0.05$. But as we can see the algorithm is able to scale even to very high noise levels which could present an advantage in industrial settings or on vehicles, or where fast feedback is not possible.

It is also interesting to see how many iterations are needed until the system is locked. For this a zoomed version of figure 5 is given in figure 6. As we can see, it only took 6 steps in this case and was repeatedly in the 5 to 12 steps range. Naturally, this is much faster than a modulation technique as we can use all of the images information on one shot. On the other hand, a wavefront sensing technique would be instantaneous but more prone to noise.

4. Reward functions

Reward functions play an important role in RL although they are only needed in the training phase. The sum of the phase differences is an obvious choice as it is very clear and offers very high contrast smooth feedback and easy comparability between optical geometries. However, it might not always be accessible or it might not be what is really wanted. Since the resulting control policy depends on the reward function, here we will discuss a few other options. A very practical reward function is power through a pinhole, as this might be what is really optimized for in a practical experiment and also something that can be measured easily. For

![Figure 5. Performance of the algorithm in the presence of noise.](image)

![Figure 6. Convergence to locked state in six steps.](image)
Figure 7. The two possibilities to obtain a donut shaped far field. (a) Resulting far field pattern, (b) near field with the two possible phase patterns indicated in red and green.

this, all we have to do is to multiply with a mask which is 1 where the hole is and 0 elsewhere and sum up the intensity as shown by equation (4).

$$R = \int I(x,y) \cdot \theta(r_0 - r(x,y)) dx dy.$$  \hspace{1cm} (4)

There is a small disadvantage in the fact that we manually need to set the position of the pinhole and the feedback might not be as sensitive to phase disturbances. This can be avoided with the reward function

$$R = -\int \frac{I(x,y)}{\max(I(x,y))} dx dy$$  \hspace{1cm} (5)

which maximizes the fractional power in the maximum. Or

$$R = \int I(x,y)^2 dx dy$$  \hspace{1cm} (6)

which prefers large intensities due to the square. However, while these metrics will in many cases realize what is really wanted, there are two dangers here: The reward function given by equation (5) relies on a single-pixel readout which could easily be faulty and the reward function (6) could have its maxima at a different configuration than what was intended. Here shifting the main lobe to also obtain more intense side lobes can lead to a larger reward depending on the actual configuration. Therefore, care has to be taken when choosing reward functions and while some modification can lead to better convergence it is of the biggest importance that the reward function describes the desired behavior.

5. Beam shaping

One very attractive reward function is to optimize the output profile to fit a particular image

$$r = -\int (I(x,y) - I(x,y)_0)^2 dx dy.$$  \hspace{1cm} (7)

This means it can be used for beam shaping without caring on what the exact phase pattern is best suited to obtain a particular pattern. We tried this by stabilizing to a donut-shaped beam in a six emitter configuration (figure 7 (a)). This is particularly interesting because there are two possibilities to obtain pattern as it can be caused by right-hand or left-hand phase (figure 7 (b)).

When we used equation (3) to specifically train for the left hand or right-hand pattern to see if it is possible to select either mode without explicitly measuring it. After training one neural network for each phase pattern we tested if it is possible to select the correct phase pattern with it by changing the neural network. The result is shown in figure 8. We did not stabilize anything the first 200 timesteps, then we used the neural network trained for left-hand phase till the 400th timestep and the neural network for right-hand phase in the last 200 timesteps. As can be seen the phase error with respect to the left hand or right-hand phase pattern is close to 0 when the appropriate neural network is selected. This means we can use this to select modes without explicitly measuring them.
6. Noise, dynamic range, and long term stability

In practical experiments, the sensor data will not be as clean as in a simulation, which naturally will impact the performance. While we cannot get a realistic quantitative estimate for the impact, nevertheless we investigated if the method can adapt to several disturbances in principle. First we added Gaussian intensity noise to the image and compared the controller performance to that of a perfect image. This only resulted in a negligible performance penalty as long as one could still recognize the shape by eye.

Furthermore, we decreased the dynamic range. As the simulated image is represented as 32-bit floating-point values, the capability of any realistic camera is certainly going to be lower. Typically somewhere in the range of 8–16 bit. There are two possibilities to use this limited dynamic range with or without auto exposure. Including auto-exposure will eliminate any power estimate at an individual, pixel while not including auto-exposure will increase the risk of not being able to resolve more fine structures in the beam shape due to under or overexposure. We tried both options with a limit to 8 bit as a conservative estimate. In the case of fixed exposure, the range had to be set carefully but when done so, both methods worked with only a small performance penalty.

There is one more challenge in real experiments related to long term drifts. As RL needs the training time if the state of the system shifts beyond the RL system's control, for example, the polarization and this leads to a significant drift of the far-field shape as well this will severely impact the convergence of RL. For this reason, it is of particular importance to keep the long term stability of such a system in mind when using this technique in practice.

7. Conclusion

We have investigated the use of RL in a simulated tiled aperture beam combining system. In this simulated environment, the trained agent is capable of controlling the input phase, based just on the far-field image. We have shown that a single image is sufficient for this and the approach scales over many geometries and emitter counts. We have shown that beam shaping is also possible and various reward functions can be used. We conclude that this method shows great promise in our simulations and as a next step should be verified experimentally.

This approach might also be applied to beam quality stabilization in general as in multimode fibers or modal instabilities in fiber amplifiers or photonic lanterns if an appropriate actuator can be found.

Funding

JST-Mirai Program Grant Number JPMJMI17A1, Japan; Japan Society for the Promotion of Science (JSPS) (JP18H01896); Ministry of Education, Culture, Sports, Science and Technology (MEXT) Quantum Leap Flagship Program (MEXT Q-LEAP).
References

[1] Fan T Y 2005 Laser beam combining for high-power, high-radiance sources IEEE J. Sel. Top. Quantum Electron. 11 567–77
[2] Klenke A, Müller M, Stark H, Kienel M, Jauregui C, Tünnermann A and Limpert J 2018 Coherent beam combination of ultrafast fiber lasers IEEE J. Sel. Top. Quantum Electron. 24 1–9
[3] Müller M, Kienel M, Klenke A, Gottschall T, Shestaev E, Pлотнер M, Limpert J and Tünnermann A 2016 1 kW 1 mJ eight-channel ultrafast fiber laser Opt. Lett. 41 3439–42
[4] Tünnermann H and Shirakawa A 2019 Deep reinforcement learning for coherent beam combining applications Opt. Express 27 24223–30
[5] Shay T M, Benham V, Baker J T, Sanchez A D, Pilkington D and Lu C A 2007 Self-synchronous and self-referenced coherent beam combination for large optical arrays IEEE J. Sel. Top. Quantum Electron. 13 480–6
[6] Hansch T and Couillaud B 1980 Laser frequency stabilization by polarization spectroscopy of a reflecting reference cavity Opt. Commun. 35 441–4
[7] Sutton R S and Barto A G 2018 Reinforcement learning: An introduction (Cambridge, MA: MIT Press)
[8] Mnih V et al 2015 Human-level control through deep reinforcement learning Nature 518 529–33
[9] Lillicrap T P, Hunt J J, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, and Wierstra D 2015 Continuous control with deep reinforcement learning (arXiv:1509.02971)
[10] Schulman J, Wolski F, Dhariwal P, Radford A, and Klimov O 2017 Proximal policy optimization algorithms (arXiv:1707.06347)
[11] Le Dortz J et al 2017 Highly scalable femtosecond coherent beam combining demonstrated with 19 fibers Opt. Lett. 42 1887–90
[12] Liang E, Liaw R, Nishihara R, Moritz P, Fox R, Goldberg K, Gonzalez J, Jordan M and Stoica I 2018 RLlib: abstractions for distributed reinforcement learning Int. Conf. on Machine Learning pp 3053–62