Deep Reinforcement Learning for Subpixel Neural Tracking

Tianhong Dai[1] TIANHONG.DAI15@IMPERIAL.AC.UK
Magda Dubois[1] MAGDA.DUBOIS.18@UCL.AC.UK
Kai Arulkumaran[1] KAILASH.ARULKUMARAN13@IMPERIAL.AC.UK
Jonathan Campbell[1] JONATHAN.CAMPBELL13@IMPERIAL.AC.UK
Cher Bass[1] C.BASS14@IMPERIAL.AC.UK
Benjamin Billot[1] BENJAMIN.BILLOT.18@UCL.AC.UK
Fatmatulzehra Uslu[1] F.USLU13@IMPERIAL.AC.UK
Vincenzo de Paola[2] VINCENZO.DEPAOLA@CSC.MRC.AC.UK
Claudia Clopath[1] C.CLOPATH@IMPERIAL.AC.UK
Anil Anthony Bharath[1] A.BHARATH@IMPERIAL.AC.UK
[1] *Department of Bioengineering, Imperial College London, London, UK*
[2] *MRC Clinical Science Centre, Faculty of Medicine, Imperial College London, London, UK*

Abstract

Automatically tracing elongated structures, such as axons and blood vessels, is a challenging problem in the field of biomedical imaging, but one with many downstream applications. Real, labelled data is sparse, and existing algorithms either lack robustness to different datasets, or otherwise require significant manual tuning. Here, we instead learn a tracking algorithm in a synthetic environment, and apply it to tracing axons. To do so, we formulate tracking as a reinforcement learning problem, and apply deep reinforcement learning techniques with a continuous action space to learn how to track at the subpixel level. We train our model on simple synthetic data and test it on mouse cortical two-photon microscopy images. Despite the domain gap, our model approaches the performance of a heavily engineered tracker from a standard analysis suite for neuronal microscopy. We show that fine-tuning on real data improves performance, allowing better transfer when real labelled data is available. Finally, we demonstrate that our model's uncertainty measure—a feature lacking in hand-engineered trackers—corresponds with how well it tracks the structure.

Keywords: tracking, tracing, neuron, axon, reinforcement learning, transfer learning

1. Introduction

Although image segmentation has received significant attention as a tool for analysing biomedical image data (Greenspan et al., 2016), it does not immediately provide geometric information. Indeed, semantic pixel-level segmentation is often an input to further analytical processes: measuring sizes, areas, or being used as inputs to global shape representations. In contrast, tracking—often implemented through Kalman filtering, particle filtering, or semi-heuristic connectivity algorithms, can provide additional structural information. In particular, tracking differs from segmentation in several crucial ways:

- Tracking establishes an *order* to locations;

- Tracking can be used to capture semantically useful properties of data (e.g. velocity of movement; length of a structure) without an explicit label being applied to each observation point;
- Algorithms for tracking (Bar-Shalom and Li, 1995; Van Trees and Bell, 2007) often include some form of model-based parameter estimation for properties of interest (e.g., Kalman filters for velocity estimation).

Tracking may also involve solving some form of correspondence, or target assignment problem: this is particularly true when multiple structures are being tracked, i.e., multiple, distinct paths exist within an image, and also when objects have gaps due to geometric factors associated with slice selection or confocal imaging. An analogy can be drawn to tracking pedestrians from video data: multiple pedestrians may need to be tracked, and occlusions from other people or objects are common. While a segmentation of an image frame would typically exclude occluded objects, a tracking algorithm would need to infer the objects' locations through time and space.

Tracking is of interest in biomedical imaging where it can be applied to analyse thin, elongated structures that might vary in apparent contrast and curvature, or have crossing and branching patterns. The structured output that can be produced by tracking is useful to quantify different aspects of the underlying geometry (e.g., number of structures/branches, crossing point locations, etc.), which is not possible with segmentation. As such, tracking algorithms have been applied to biomedical datasets with thin structures, for example: neurons (Meijering, 2010; Peng et al., 2010, 2015; Acciai et al., 2016; Poulin et al., 2017), blood vessels (Fraz et al., 2012; Kumar et al., 2015), and muscle fibres (Farris and Lichtwark, 2016).

To alleviate the need for hand-engineering trackers for different biomedical image datasets, we first formulate the task of tracing paths along the centrelines of elongated structures as a reinforcement learning (RL) problem, and then explore the use of deep RL (DRL) to train deep convolutional neural networks (CNNs) to learn tracking policies (Zhang et al., 2018). This removes the need for explicit appearance (observation) models and state evolution models, and additionally enables potentially richer objectives to be optimised.

We extend prior work in several ways. Firstly, we utilise a continuous action space, in contrast to prior work using DRL in biomedical imaging (Ghesu et al., 2016; Maicas et al., 2017; Liao et al., 2017; Krebs et al., 2017; Alansary et al., 2018; Zhang et al., 2018; Al and Yun, 2018; Ghesu et al., 2019), to perform subpixel tracking. Secondly, unlike Zhang et al. (2018), we address the challenge of limited training data, which is common in biomedical settings; we train our model on a simple synthetic dataset and test it on an axonal mouse cortex dataset (Bass et al., 2017), which contains many crossing elongated structures. The results of our tracker, which has to perform transductive (Pan et al., 2010) or "zero-shot" transfer to the microscopy data, can be seen in Figure 1. Despite this, our model's performance approaches that of the current standard in the field, the Vaa3D tracker (Peng et al., 2010). By fine-tuning on the real data, which is viable with a small amount of labelled data in a mainly unlabelled dataset, we can improve performance even further. Finally, we demonstrate that the entropy of our model's outputs—a measure of uncertainty—corresponds with how well it stays on track. Such a property is valuable in biomedical contexts, and is often lacking from many trackers, including Vaa3D's. Our work

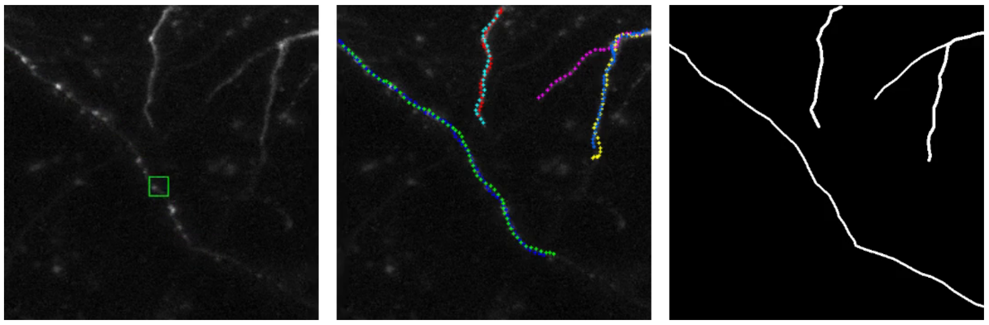


Figure 1: The operation of our learned tracker (left) on a microscopy image of several axons (Bass et al., 2017). After starting the tracker from manually labelled end-points, we can construct traces of all found axons, including branches (centre). A manual segmentation of the axons (right) is provided for comparison.

represents another step towards learning general trackers for biomedical images, and we have open sourced our code[1] and data[2] to support further research in this direction.

2. Background

2.1. Reinforcement Learning

RL is a branch of machine learning in which the objective is to learn to make an optimal sequence of decisions in order to maximise a reward (Sutton and Barto, 1992). In our case, the task is to trace the centreline of a biological structure from one end to another. Given the wide variety in the appearance of these structures, as well as the imaging conditions, our aim is to develop a learning agent that can be trained on the data in question, rather than having to manually tune a fixed tracking algorithm on each new dataset. We shall now explain RL more generally, as well as our specific algorithmic choices.

In RL, an agent inhabits an environment, and makes a sequence of decisions based on what it observes. At every timestep t, the agent receives the current state of the environment, s_t, and chooses an action, a_t, according to its policy π—a probability distribution that maps states to actions. As a result of taking an action, the agent receives a new state together with a scalar reward r_{t+1}, which gives it information about its performance. The goal of the agent is to maximise its expected return, $\mathbb{E}[R]$, in episodes of length T, where $R = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$. Here, $\gamma \in [0, 1]$ is a discount factor which can be tuned to prioritise immediate rewards over more distant ones.

To solve our RL problem, we use an actor-critic algorithm, which combines learning a policy (actor) and value function (critic) (Sutton and Barto, 1992). Moreover, we use DRL, modelling both the policy and value functions with CNNs.[3] The use of CNNs in RL agents allows them to learn directly from images, with successful applications including real-world visual navigation (Arulkumaran et al., 2017).

1. https://bitbucket.org/bicv/axon_tracking_with_rl/
2. https://www.zenodo.org/record/1182487
3. The CNN architectures can be found in Appendix A.

In order to achieve *subpixel* accuracy when tracing centrelines, the agent should ideally output continuous (real-valued) actions that can be mapped to subpixel coordinates. We therefore use proximal policy optimization (PPO) (Schulman et al., 2017), a state-of-the-art actor-critic algorithm which supports continuous action policies. Its main benefit is the use of an adaptive penalty on the amount by which the policy can change during an update, which results in more stable training than many other RL algorithms.

The original PPO algorithm used Gaussian distributions, which have infinite support (Schulman et al., 2017). In practice, we may want to constrain the action space, for example to prevent the agent from traversing many pixels in one action. Rather than penalising or thresholding large actions, which can adversely bias learning, we instead use beta distributions, which have finite support and have empirically been shown to improve the convergence of DRL algorithms on a range of problems (Chou et al., 2017).

Actor-critic algorithms make use of a learned value function to reduce the variance of the policy updates (Schulman et al., 2016). We use a particular form of variance reduction technique known as generalised advantage estimation (GAE) (Schulman et al., 2016), which introduces a small bias in return for extra variance reduction.

In actor-critic methods, the value function is only needed during training. Pinto et al. (2017) introduced the asymmetric actor-critic algorithm, in which the critic is given extra information that is available only during training, which can improve the learning of the true value function. Consequentially, the actor then benefits from less biased policy updates. In our setup, we achieve this by giving the critic access to the "ground truth" (the centreline, which acts as a visual representation of the reward), which allows it to learn the underlying value function more easily. The actor does not view the ground truth; and once trained, it operates directly upon on the image data, with no explicit segmentation.

2.2. Biomedical Imaging

There are several areas within the field of biomedical imaging that are related to our work. We will briefly cover research on thin structures within biomedical images (e.g., axons, neurons and vessels), tracking, and other uses of DRL for image analysis tasks.

2.2.1. Thin Structures

The analysis of elongated anatomical structures such as neurons, retinal vasculature, and muscle fibres, are important in the fields of medicine, physiology and neuroscience for diagnosis, and for the study of biological processes. These thin structures have varying properties, due to the differences in imaging techniques (microscopy, ultrasound), conditions (lighting, noise), and due to the underlying biological variability. For example, whole neurons have cell bodies, as well as dendrites and axons containing synapses which are blob-like in appearance, while retinal vasculature has a tree like structure, often with many branches (Fraz et al., 2012). Much of the prior work has focused on using segmentation algorithms for the analysis of these datasets. Segmentation of retinal vasculature has been extensively studied, and we refer the reader to Fraz et al. (2012) and Kirbas and Quek (2003) for detailed reviews of the topic. More recent works have used deep neural networks to segment thin structures in vessels (Maninis et al., 2016), muscle fibres (Farris and Lichtwark, 2016; Xie et al., 2016; Cunningham et al., 2017), whole neurons (Zhou et al., 2018; Li et al., 2017;

Liu et al., 2018), and in axons (Bass et al., 2019). Of particular relevance is the work of Li et al. (2017), which involved both segmentation and tracking. They used a CNN for the segmentation of whole neurons, followed by a tracking algorithm to extract a graph structure from the segmentation map. While successful, this type of approach relies on an accurate segmentation for tracking to succeed.

2.2.2. Tracking

Tracking in biomedical images has been used for a range of applications, including tracing elongated structures such as neurons (Meijering, 2010; Skibbe et al., 2019), vessels (Fraz et al., 2012), and muscle fibres (Farris and Lichtwark, 2016). Tracking can be used either with or without prior segmentation in situations where (i) one wants to quantify the tracked structure, (ii) the proportion of pixels representing the structure of interest is small (Helmstaedter et al., 2008) and (iii) there are branches, terminations or obscuring structures, such as blood vessels (Fraz et al., 2012). The path established by a tracker can be used to order and capture quantitative measures about morphology of entities, such as width, direction, the presence and number of branches, information that requires additional processing if segmentation is used. Moreover, tracking methods can identify branching points over the course of tracing and they can maintain the identities of branches emerging from the branching points, providing information on topology and connectivity.

2.2.3. Deep Reinforcement Learning

The combination of deep neural networks with reinforcement learning (DRL) has been successfully utilised across a range of applications in the last few years (Arulkumaran et al., 2017), including biomedical imaging (Ghesu et al., 2016; Maicas et al., 2017; Liao et al., 2017; Krebs et al., 2017; Alansary et al., 2018; Zhang et al., 2018; Al and Yun, 2018; Ghesu et al., 2019). These combine both feature learning (as opposed to utilising hand-engineered appearance models) with a general optimisation objective, formulated as a sequential decision problem in order to fit into the RL framework. Prior works have included applications to landmark detection (where the agent is similarly "embodied" in the image and must find a specific structure) (Ghesu et al., 2016; Maicas et al., 2017; Al and Yun, 2018; Ghesu et al., 2019), view planning (finding optimal 2D views in 3D images for downstream tasks) (Alansary et al., 2018), and image registration (aligning images to the same coordinate system) (Liao et al., 2017; Krebs et al., 2017). While these applications allow the agent to take any path to the solution, tracking requires the path to be as close as possible to the ground truth path of the underlying (in our case, anatomical) structure at every point. Having an underlying path allows for a denser reward signal, but also leaves less room for failure. Zhang et al. (2018) previously proposed the use of DRL for centreline tracing for blood vessels. One of the major differences is their use of a discrete action space, which limits their ability to make subvoxel traces.[4] In addition, they are able to train directly on hundreds of real labelled images, so do not have to address a transfer learning problem. Finally (with the exception of Al and Yun (2018) who also use an actor-critic approach but still with discrete actions), all of these prior works have been based on the deep Q-network (DQN) (Mnih et al., 2015) or variants thereof, and are hence restricted to discrete action spaces;

4. We also formulate a different reward function to specifically account for taking subpixel movements.

whereas we utilise DRL with a continuous action space for biomedical imaging applications. Additionally, the output of DQNs are "Q-values" as opposed to probability distributions, where the latter allows us to directly provide uncertainty estimates via the entropy of the policy.

3. Subpixel Neural Tracking via Reinforcement Learning

3.1. Environment

We now discuss how tracing centrelines in medical images can be formulated as an RL problem. This includes the environment, the state and action spaces and the reward function.
Environment: Our environment is based on 2D greyscale images of a neuron; in RL terminology we use one image per "episode", such that the agent receives a new neuron to track every episode. For our experiments, we use synthetic and microscopy images. In both cases the background is noisy with a low average intensity, and the neuronal structures are depicted by brighter pixels. We treat the agent as being "embodied" in the environment, which means that it is positioned within the image. The agent starts at a predefined position—one end of an axon—and moves until it finds another end. If the agent does not find another end, we terminate the episode after 200 timesteps. The environment generation is highly stochastic, but with selectable degrees of complexity.
State space: Rather than using the whole image as input, we provide an egocentric, multiscale view to the agent, with all views at 11×11px. The view comprises of one window at full resolution (11×11px) and one 21×21px window downscaled via bilinear interpolation to 11×11px.[5] In order to give temporal context to the agent, we also provide the historical path—a view which shows the previously visited pixels around the agent's current position. The (asymmetric) critic also receives a view containing the centreline. As the agent's location, and hence the centre of the views, is specified with subpixel accuracy, we use bilinear interpolation to provide a correctly centred viewpoint to the agent. We zero pad all images when the view extends beyond the edge of the original image. Finally, to provide further temporal context, we concatenate all the different types of views with the corresponding views from the 3 previous timesteps (Mnih et al., 2015). The full state for the agent is visualised in Figure 2.
Action space: Fine structures, like axons, can be smaller than the pixel size of the image, due to discretisation in the imaging process. Furthermore, data acquisition can occur at different resolutions. To account for this, our aim is to achieve subpixel tracking. There are two components to attaining subpixel accuracy. First, rather than using discrete control, we use continuous control, with actions moving the position of the agent in 2D space. This means that the estimated position as the result of any action comes in the form of floating point coordinates that potentially (indeed, usually) are in between pixel centres. The second component relates to the reward function, discussed below.
Reward function: Our goal is for the agent to follow the centreline of an axon, which needs to be expressed via an appropriate reward function. The base reward is the average integral of intensity between the agent's current and next location. To achieve subpixel

5. As our synthetic data is slightly lower resolution than our real data, we use $1.5\times$ the window size on the real data, downscaled to 11×11px.

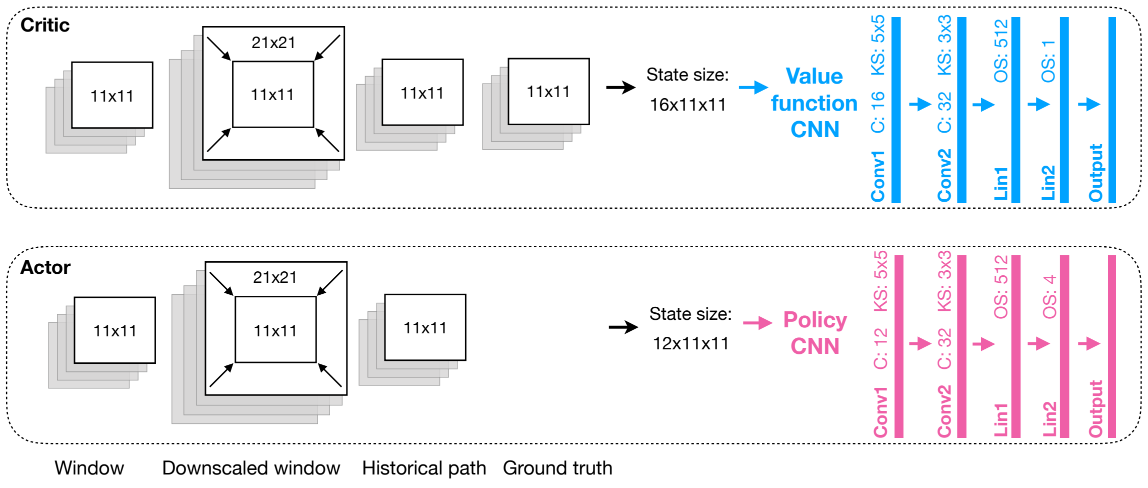


Figure 2: Inputs and architectures of the actor and the critic. All views are centered on the agent at every step, with dimensions given in pixels. The window and the downscaled window contain the pixel intensity values from the neuron image; the historical path contains only information about the agent's previous positions and the ground truth. The state contains 4 timesteps of this information. Both CNNs contain 2 convolutional (Conv) and 2 fully-connected (Lin) layers. We also show the number of channels (C), kernel size (KS) and output size (OS).

accuracy, the spatial distribution of image intensities is resampled at subpixel locations, using bilinear interpolation, to calculate the reward function using the integral of intensity along straight line segments. As extra heuristics that we found to be empirically useful, we also provide a negative reward if the agent does not move, and also penalise switching directions more than once (defined as an action that is $> 90°$ from the previous action).[6]

3.2. Agent

The policy and value function of the agent are represented by separate CNNs, and are depicted in Figure 2, along with their respective inputs. The output of the actor network is a set of parameters for two independent beta distributions, from which actions (displacements in the x and y coordinates of the agent) can be sampled.[7] During training we sample actions, but during testing we use the means of the distributions, which results in a deterministic policy for evaluation. As the support of the beta distribution is finite, we map samples from the policy $\in [0, 1]$ to pixel displacements $\in [-4, 4]$ in the environment. The output of the critic network is a single value representing the value function.

Both networks are updated according to the PPO algorithm (Schulman et al., 2017) with discount $\gamma = 0.99$, PPO clipping value $\epsilon = 0.2$ and GAE (Schulman et al., 2016)

6. Reward pseudocode can be found in Appendix B.
7. We restrict the output parameters to be greater than 1 so that the beta distributions are unimodal.

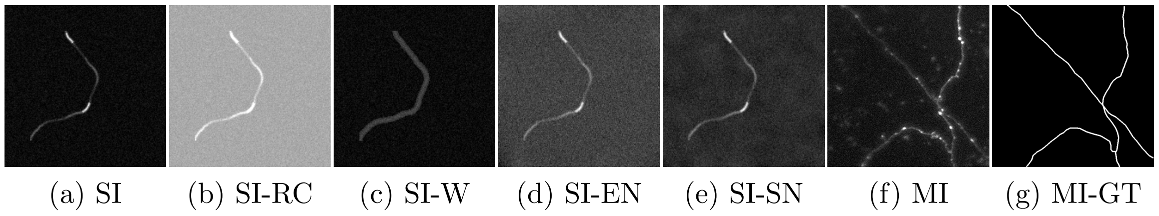


Figure 3: Examples of the 7 datasets used: synthetic (SI), synthetic reduced contrast (SI-RC), synthetic wide (SI-W), synthetic extra noise (SI-EN), synthetic structured noise (SI-SN), and microscopy images (MI). (a) is a synthetic image with the standard settings (background and axon intensities + Gaussian noise) that we use for pretraining the tracker for the MI dataset. (b), (c), (d) and (e) are synthetic images with different simulator settings. (f) was collected from a mouse somatosensory cortex using two-photon microscopy (Bass et al., 2017) with ground truth (GT) (g) labelled manually.

eligibility trace value $\lambda = 0.95$. After collecting a batch of 32 episodes, each network is updated 10 times within PPO's internal loop. We use the Adam optimiser (Kingma and Ba, 2015) with a learning rate of 0.0005 and an L2 weight decay factor of 0.0003.[8]

4. Experiments

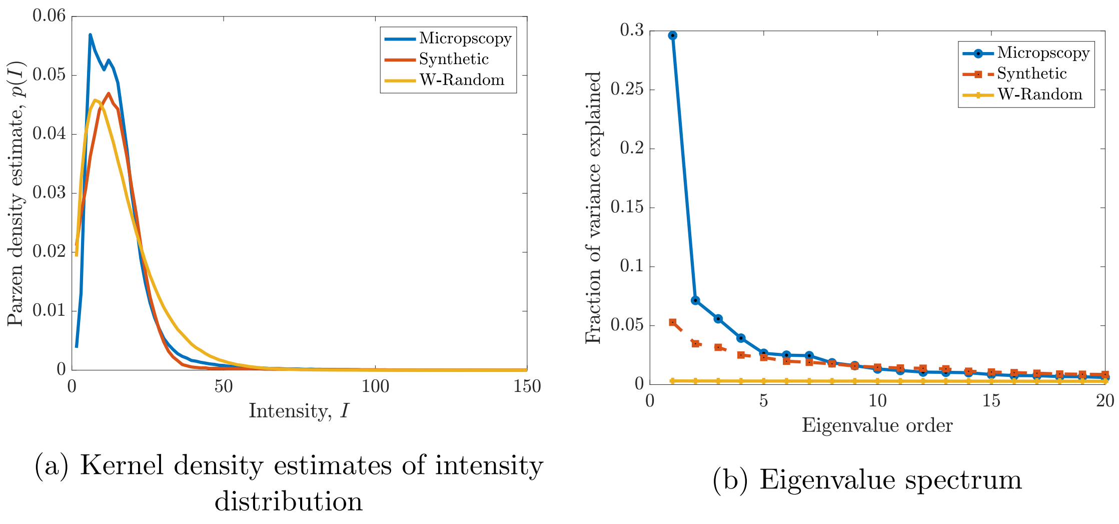
4.1. Datasets

We evaluated our DRL tracker on several synthetic datasets and a microscopy dataset (Figure 3). For all datasets we manually specified all start points (required for Vaa3D (Peng et al., 2010) and our DRL tracker; see subsection 4.2).

Synthetic datasets: Real, labelled data is often difficult to obtain (due to paucity of data, noise, etc.), and so we built a simulator to generate artificial images for training and validation. We simulated single axons by fitting polynomial splines to constrained random walks through 2D space, and adding Gaussian noise to the background. We tuned the intensity and noise settings as shown in Figure 3a to pretrain our tracker for the real data, but also trained trackers successfully on other settings, showing that our algorithm generalises to these particular conditions: lower contrast (Figure 3b), wider structures (Figure 3c), extra noise (Figure 3d) and structured (Poisson) noise (Figure 3e).

Microscopy dataset: In the second stage, we tested our tracker on a mouse cortical axon dataset (Bass et al., 2017). We took a subset of 20 2D images (produced from a 3D stack), and manually produced corresponding labels.[9] To generate the 2D images, we used a max projection of the 3D stack. The labels are in the form of ground truth binary images of the same size, in which the corresponding axons were segmented, and axon centerlines labelled manually using the Vaa3D software (Peng et al., 2010). We compare kernel density estimates and eigenvalue spectra of the synthetic (SI) and real images (MI) in Figure 4. As

8. Training pseudocode can be found in Appendix C.
9. Note that producing centreline labels is a time-consuming process requiring an expert, restricting the amount of labelled data that we were able to procure.



(a) Kernel density estimates of intensity distribution

(b) Eigenvalue spectrum

Figure 4: Quantitative comparison of synthetic and real microscopy images. (a) Kernel density estimates of pixel intensities for microscopy data vs synthetic; also shown, samples drawn from spatial white noise (W-Random) with a gamma distribution approximately matching that of real images. (b) The top 20 eigenvalue components of real data, synthetic environment, and that of white noise approximately matching the distribution of the real data. See Appendix D.1 for details.

shown in Figure 4a, the SI data matches the decay in the pixel intensity values, but does not fit the low intensity pixels as well. Figure 4b demonstrates that the SI data has some level of spatial structure, but is closer to noise in comparison to the MI data.

4.2. Training and Testing

Training for each synthetic dataset was performed on 32,000 synthetic images.[10] For each synthetic dataset we validated our hyperparameters on a held-out set of 3,600 synthetic images, for a 9:1 training/validation split. We then tested our best model trained on the SI dataset on the 20 microscopy images. For our fine-tuned model, due to the lack of labelled data we used a k-fold method in which we fine-tuned the original model on 15 images and tested on the remaining 5, repeated on 4 subsets of the data.

Two measures were used to assess performance: *coverage* and *mean absolute error*. The coverage is the percentage of the axons (keypoints defined by the labels) on which the traced points were within 3px, while the error is the average perpendicular distance between axon and trace keypoints.[11] These metrics complement each other, as coverage quantifies how robust a tracker is (e.g., in the presence of sharp bends), while accuracy is the end goal. We compared our performance against that of trackers implemented in the Vaa3D software

10. Training on each synthetic dataset takes less than 5 hours on a GeForce 1080Ti.

11. If any tracker has an error over 11px for more than 5 consecutive timesteps we consider tracking lost and exclude these periods from our average distance error measure; we include raw results in Figure 7.

Table 1: Validation performance: coverage (%) and average error ±1 standard deviation (px) of DRL trackers, trained on different sets of synthetic images and validated on corresponding simulator settings.

SI		**SI-RC**		**SI-W**		**SI-EN**		**SI-SN**	
Cov.	Error	Cov.	Error	Cov.	Error	Cov.	Error	Cov.	Error
93.37	0.64 ± 0.88	94.13	0.68 ± 0.88	94.62	0.39 ± 0.82	96.18	0.53 ± 0.66	94.72	0.59 ± 0.76

(Peng et al., 2010): both the default Vaa3D neuron reconstruction algorithm, which also requires start and end points as inputs, as well as the APP2 neuron tracer (Xiao and Peng, 2013), which does not require start and end points.

The coverage and error of the DRL tracker is very high and very low, respectively, on the synthetic validation sets, with little difference between the different image conditions (Table 1). The performance of the DRL tracker trained on the standard synthetic dataset (SI) drops when it is then applied to the microscopy dataset (84.10% / 1.88±2.23px). This is not too far from the performance of Vaa3D, which is specialised for this kind of data (92.26% / 0.89±0.84px), which is promising given that our tracker was never trained on real data, and does not incorporate any prior knowledge about microscopy data. APP2 performs the worst (81.82% / 1.61±2.82px), highlighting the difficulty of performing endpoint localisation as well as tracing. We can improve coverage by fine-tuning on a very small amount of labelled data (89.08% / 1.82±2.13px), suggesting that our two-phase training could be viable when a moderate amount of labelled data is available, or if an existing, hand-engineered tracker is not available at all. Average results are available in Table 2 with the per-image summary statistics available in Table 5 and the distribution of errors available in Figures 6 and 7.

As we use a stochastic policy, we can characterise the *uncertainty* of the DRL tracker by evaluating its entropy at any state.[12] There is a significant difference between the average entropy within (< 3px) and outside (≥ 3px) the threshold: 0.91 on average for the DRL tracker ($p < 0.00001$, paired t-test), and 0.78 for the fine-tuned DRL tracker ($p < 0.00001$, paired t-test)[13], indicating a quantitative increase in the trackers' uncertainty if they stray from an axon. The ability to extract such a value is in contrast to traditional trackers such as Vaa3D, which typically do not quantify their uncertainty.

5. Conclusion

We proposed a new approach to tracking thin biological structures, such as axons and blood vessels, that is capable of producing subpixel-level traces. This first involved formulating tracking such structures as a reinforcement learning problem (Zhang et al., 2018). We then introduced a tracker based on a combination of state-of-the-art deep reinforcement learning (DRL) techniques (Schulman et al., 2017; Chou et al., 2017; Schulman et al., 2016; Pinto et al., 2017), and exposed it to an environment which simulates the physical processes of imaging. Our DRL tracker was close to the performance, with respect to coverage, of the

12. Note that the entropy of the beta distribution is upper-bounded by 0.
13. See Table 6 for the per-image breakdown.

Table 2: Test performance: coverage (%) and average error ±1 standard deviation (px) of trackers, averaged over all microscopy images. DRL represents the performeance of the agent trained on SI data alone, and DRL (FT) represents the performance of the DRL agent after fine-tuning on the MI data.

DRL		**DRL (FT)**		**APP2**		**Vaa3D**	
Cov.	Error	Cov.	Error	Cov.	Error	Cov.	Error
84.10	1.88±2.23	89.08	1.82±2.13	81.82	1.61±2.82	92.26	0.89±0.84

current standard for tracing neurons (Peng et al., 2010), despite only being trained on simpler, synthetic data. Further, this is achieved without an explicit segmentation being applied as a pre-processing step.

We were able to improve coverage performance by fine-tuning the tracker on a small amount of real data, which makes our method viable for semi-supervised settings. A promising direction for future work is to incorporate synthetic data from conditional generative models during training in order to improve the realism of the synthetic data, and hence performance on real data; for example, the CapsPix2Pix model (Bass et al., 2019), from which synthetic images were utilised to improve the performance of segmentation models trained on the same microscopy dataset that we used in our work (Bass et al., 2017).

Finally, we were able to extract a quantitative measure of uncertainty, which corresponded to how well the tracker performed. While it appears that the uncertainty is well-calibrated, i.e., the values are significantly different when the agent is on- and off-track, the distribution of values also differ between images. Developing automated methods that stop tracking in test images using entropy is an interesting avenue for future work.

Our proposed method can naturally be applied to tracking elongated structures in other types biomedical images, and, furthermore, could serve as a building block for future research aimed at 3D subpixel tracking of elongated structures, for boundary trackers, and for other tasks that are currently difficult to fully automate. On the other hand, this also opens up a challenging new task for testing DRL algorithms, which are typically evaluated on video games with simple graphics (Bellemare et al., 2013) or simple control tasks with symbolic inputs (Brockman et al., 2016).

Acknowledgments

We would like to thank Ryutaro Tanno for general feedback and Amir Alansary for help with relevant literature. We would like to also acknowledge funding from the Samsung Global Research Outreach program, the Department of Bioengineering, Imperial College London, and the Engineering and Physical Sciences Research Council [grant number EP/L016737/1].

References

Ludovica Acciai, Paolo Soda, and Giulio Iannello. Automated neuron tracing methods: an updated account. *Neuroinformatics*, 14(4):353–367, 2016.

Walid Abdullah Al and Il Dong Yun. Partial policy-based reinforcement learning for anatomical landmark localization in 3d medical images. *arXiv preprint arXiv:1807.02908*, 2018.

Amir Alansary, Loic Le Folgoc, Ghislain Vaillant, Ozan Oktay, Yuanwei Li, Wenjia Bai, Jonathan Passerat-Palmbach, Ricardo Guerrero, Konstantinos Kamnitsas, Benjamin Hou, et al. Automatic view planning with multi-scale deep reinforcement learning agents. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 277–285. Springer, 2018.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE SPM*, 34(6):26–38, 2017.

Yaakov Bar-Shalom and Xiao-Rong Li. *Multitarget-multisensor tracking: principles and techniques*, volume 19. YBs London, UK:, 1995.

Cher Bass, Pyry Helkkula, Vincenzo De Paola, Claudia Clopath, and Anil Anthony Bharath. Detection of axonal synapses in 3D two-photon images. *PLoS ONE*, 12(9):1–18, 2017. ISSN 19326203. doi: 10.1371/journal.pone.0183309.

Cher Bass, Tianhong Dai, Benjamin Billot, Kai Arulkumaran, Antonia Creswell, Claudia Clopath, Vincenzo De Paola, and Anil Anthony Bharath. Image synthesis with a convolutional capsule generative adversarial network. In *Medical Imaging with Deep Learning*, 2019.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *JAIR*, 47:253–279, 2013.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Po-Wei Chou, Daniel Maturana, and Sebastian Scherer. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *ICML*, pages 834–843, 2017.

Ryan Cunningham, Peter Harding, and Ian Loram. Deep residual networks for quantification of muscle fiber orientation and curvature from ultrasound images. In *Annual Conference on Medical Image Understanding and Analysis*, pages 63–73. Springer, 2017.

Dominic James Farris and Glen A Lichtwark. Ultratrack: Software for semi-automated tracking of muscle fascicles in sequences of b-mode ultrasound images. *Computer methods and programs in biomedicine*, 128:111–118, 2016.

Muhammad Moazam Fraz, Paolo Remagnino, Andreas Hoppe, Bunyarit Uyyanonvara, Alicja R Rudnicka, Christopher G Owen, and Sarah A Barman. Blood vessel segmentation methodologies in retinal images–a survey. *Computer Methods and Programs in Biomedicine*, 108(1):407–433, 2012.

Florin C Ghesu, Bogdan Georgescu, Tommaso Mansi, Dominik Neumann, Joachim Hornegger, and Dorin Comaniciu. An artificial agent for anatomical landmark detection in medical images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 229–237. Springer, 2016.

Florin-Cristian Ghesu, Bogdan Georgescu, Yefeng Zheng, Sasa Grbic, Andreas Maier, Joachim Hornegger, and Dorin Comaniciu. Multi-scale deep reinforcement learning for real-time 3d-landmark detection in ct scans. *IEEE transactions on pattern analysis and machine intelligence*, 41(1):176–189, 2019.

Hayit Greenspan, Bram Van Ginneken, and Ronald M. Summers. Guest Editorial Deep Learning in Medical Imaging: Overview and Future Promise of an Exciting New Technique. *IEEE Trans. Medical Imaging*, 35(5):1153–1159, 2016. ISSN 1558254X. doi: 10.1109/TMI.2016.2553401.

Moritz Helmstaedter, Kevin L Briggman, and Winfried Denk. 3d structural imaging of the brain with photons and electrons. *Current opinion in neurobiology*, 18(6):633–641, 2008.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Cemil Kirbas and Francis KH Quek. Vessel extraction techniques and algorithms: a survey. In *Proc. Bioinformatics and Bioengineering (BIBE)*, pages 238–245. IEEE, 2003.

Julian Krebs, Tommaso Mansi, Hervé Delingette, Li Zhang, Florin C Ghesu, Shun Miao, Andreas K Maier, Nicholas Ayache, Rui Liao, and Ali Kamen. Robust non-rigid registration through agent-based action learning. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 344–352. Springer, 2017.

Rahul Prasanna Kumar, Fritz Albregtsen, Martin Reimers, Bjørn Edwin, Thomas Langø, and Ole Jakob Elle. Blood vessel segmentation and centerline tracking using local structure analysis. In *6th European conference of the international federation for medical and biological engineering*, pages 122–125. Springer, 2015.

Rongjian Li, Tao Zeng, Hanchuan Peng, and Shuiwang Ji. Deep learning segmentation of optical microscopy images improves 3-d neuron reconstruction. *IEEE transactions on medical imaging*, 36(7):1533–1541, 2017.

Rui Liao, Shun Miao, Pierre de Tournemire, Sasa Grbic, Ali Kamen, Tommaso Mansi, and Dorin Comaniciu. An artificial agent for robust image registration. In *AAAI Conference on Artificial Intelligence*, 2017.

Min Liu, Huiqiong Luo, Yinghui Tan, Xueping Wang, and Weixun Chen. Improved v-net based image segmentation for 3d neuron reconstruction. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 443–448. IEEE, 2018.

Gabriel Maicas, Gustavo Carneiro, Andrew P Bradley, Jacinto C Nascimento, and Ian Reid. Deep reinforcement learning for active breast lesion detection from dce-mri. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 665–673. Springer, 2017.

Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Arbeláez, and Luc Van Gool. Deep retinal image understanding. In *International conference on medical image computing and computer-assisted intervention*, pages 140–148. Springer, 2016.

Erik Meijering. Neuron tracing in perspective. *Cytometry Part A*, 77(7):693–704, 2010.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei a Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. ISSN 0028-0836. doi: 10.1038/nature14236.

Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Hanchuan Peng, Zongcai Ruan, Fuhui Long, Julie H. Simpson, and Eugene Myers. V3d enables real-time 3d visualization and quantitative analysis of large-scale biological image data sets. *Nature Biotechnology*, 28(4):348–353, 2010. doi: 10.1038/nbt.1612.

Hanchuan Peng, Michael Hawrylycz, Jane Roskams, Sean Hill, Nelson Spruston, Erik Meijering, and Giorgio A. Ascoli. BigNeuron: Large-Scale 3D Neuron Reconstruction from Optical Microscopy Images, 2015. ISSN 10974199.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

Philippe Poulin, Marc-Alexandre Cote, Jean-Christophe Houde, Laurent Petit, Peter F Neher, Klaus H Maier-Hein, Hugo Larochelle, and Maxime Descoteaux. Learn to track: Deep learning for tractography. In *MICCAI*, pages 540–547. Springer, 2017.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv*, pages 1–9, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Henrik Skibbe, Marco Reisert, Ken Nakae, Akiya Watakabe, Junichi Hata, Hiroaki Mizukami, Hideyuki Okano, Tetsuo Yamamori, and Shin Ishii. Patprobabilistic axon tracking for densely labeled neurons in large 3-d micrographs. *IEEE transactions on medical imaging*, 38(1):69–78, 2019.

R.S. Sutton and A.G. Barto. *Reinforcement Learning*. Springer US, Boston, MA, sep 1992. ISBN 978-1-4613-6608-9. doi: 10.1007/978-1-4615-3618-5.

Harry L Van Trees and Kristine L Bell. Bayesian bounds for parameter estimation and nonlinear filtering/tracking. *AMC*, 10:12, 2007.

Hang Xiao and Hanchuan Peng. App2: automatic tracing of 3d neuron morphology based on hierarchical pruning of a gray-weighted image distance-tree. *Bioinformatics*, 29(11): 1448–1454, 2013.

Yuanpu Xie, Zizhao Zhang, Manish Sapkota, and Lin Yang. Spatial clockwork recurrent neural network for muscle perimysium segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 185–193. Springer, 2016.

Pengyue Zhang, Fusheng Wang, and Yefeng Zheng. Deep reinforcement learning for vessel centerline tracing in multi-modality 3d volumes. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 755–763. Springer, 2018.

Zhi Zhou, Hsien-Chi Kuo, Hanchuan Peng, and Fuhui Long. Deepneuron: an open deep learning toolbox for neuron tracing. *Brain informatics*, 5(2):3, 2018.

Appendix A. Neural Network Architectures

Tables 3 and 4 contain the network architectures for the policy and value networks, respectively. We use ReLU nonlinearities between all hidden layers.

Table 3: Actor/policy network. The output of the network is two pairs of α and β parameters for two beta distributions corresponding to displacements in the x and y coordinates. We use a softplus nonlinearity and add 1 to the outputs of the final layer to ensure that these parameters are > 1; this is done so that the beta distributions are always unimodal.

Layer type	Input channels	Input size	Kernel size	Stride	Padding	Output channels	Output size
Convolution	12	11×11	5×5	1	2	32	11×11
Convolution	32	11×11	3×3	1	1	32	13×13
Linear	-	5408	-	-	-	-	512
Linear	-	512	-	-	-	-	4

Table 4: Critic/value network. The output of the network is a single real number representing the state's value function.

Layer type	Input channels	Input size	Kernel size	Stride	Padding	Output channels	Output size
Convolution	16	11×11	5×5	1	2	32	11×11
Convolution	32	11×11	3×3	1	1	32	13×13
Linear	-	5408	-	-	-	-	512
Linear	-	512	-	-	-	-	1

Appendix B. Reward Function

Algorithm 1 DRL Axon Tracking Algorithms - Reward Calculation

```
 1: Input: current agent position $p_t$, current action $a_t$, previous action $a_{t-1}$, ground truth
    centreline image $I_G$, counter $k$
 2: Output: reward $r_{t+1}$
 3: Next agent position $p_{t+1} \leftarrow p_t + a_t$
 4: Distance travelled $d \leftarrow \|p_t - p_{t-1}\|$
 5: if $d == 0$ then
 6:    $r_{t+1}$ = -1
 7: else
 8:    Sample 100 points $S$ between $p_t$ and $p_{t+1}$
 9:    Reward $r_{t+1} \leftarrow \sum_{s=1}^{|S|} I_G(x_s, y_s)$
10:    if angle between $a_{t-1}$ and $a_t > 90°$ then
11:       $k \leftarrow k + 1$
12:    end if
13:    if $k \pmod 2 == 1$ then
14:       $r_{t+1} \leftarrow -r_{t+1}$
15:    end if
16: end if
    return $r_t$
```

Appendix C. Training Algorithm

Algorithm 2 DRL Axon Tracking Algorithms - Training

```
1: Set the batch size $M$ and the max length of each episode $N$
2: Initialize the actor-network $A(s_a|\theta^a)$
3: Initialize the critic-network $C(s_c|\theta^c)$
4: repeat
5:    for $i = 1$ to $M$ do
6:       Initialize the position of agent $p_0$
7:       Initialize the counter $k \leftarrow 0$
8:       Initialize the initial unit direction vector $v_0$
9:       Initialize the initial state for actor-network $s_a$
10:      Initialize the initial state for critic-network $s_c$
11:      for $t = 1$ to $N$ do
12:         Select the action $a_t = A(s_a|\theta^a)$;
13:         Input the action $a_t$ into the simulator and get $s'_a$, $s'_c$, $r_t$, terminal;
14:         Store the transition ($s_a$, $s_c$, $r_t$, terminal);
15:         if terminal then
16:            break;
17:         end if
18:         $s_a \leftarrow s'_a$, $s_c \leftarrow s'_c$;
19:      end for
20:   end for
21:   Update the actor-network $A(s_a|\theta^a)$ by using PPO
22:   Update the critic-network $C(s_c|\theta^c)$ by using PPO
23: until $A(s_a|\theta^a)$ and $C(s_c|\theta^c)$ are converged
```

Appendix D. Further Experimental Results

D.1. Plausibility of Synthetic Environment

The use of synthetic environments for training DRL agents is common in robotics and autonomous driving, but less so for image interpretation. In any use of simulation environments, the size of the "reality gap" is a key factor in determining the ability to transfer the agent into a real environment or task. Here, we describe the experiments to compare the synthetic environment with real microscopy data.

We drew 20 images from the synthetic environments, and compared pixel intensity statistics and second-order spatial statistics through eigen spectral analysis of the covariance matrices estimated from randomly selected image patches. We also applied this analysis to white noise spatial fields, and to 20 real microscopy images.

Firstly, from each of the 20 synthetic and 20 microscopy images analysed, we randomly selected 200 25×25 pixel spatial patches, yielding 4,000 patches. We then applied a kernel density estimate to this data, using a Gaussian kernel of bandwidth 0.5 to produce estimates of image intensity in Figure 4a.

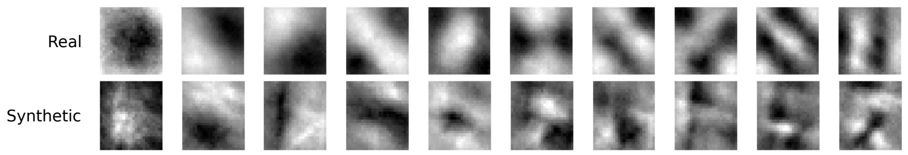


Figure 5: Eigenimages produced by finding the most important 10 eigenvalues for real image patches and for synthetic patches. Note the similarity of the first 5 eigenimages in the top and bottom rows, though there is a polarity change visible in many patches, and clear differences for eigenimages 6-10.

Once we had the histogram of pixel intensities for real images, we fit a gamma distribution to the intensity data, obtaining tight confidence intervals for the parameters (shape parameter 1.934 ± 0.003 with 95% confidence, and scale parameter of 7.699 ± 0.014 with 95% confidence). We then used these parameters to synthesise gamma-distributed white noise fields.

We also estimated the covariance matrix for each of the 3 sets of 4,000 image patches. The eigenvalue spectra (eigenvalue normalised by sum of absolute eigenvalues) for the top 20 components are shown in Figure 4b. The associated top 10 eigenimages corresponding to the microscopy and synthetic environments are presented in Figure 5. The eigenimages corresponding to white noise are essentially noise fields, and are not shown.

D.2. Error analysis during tracking

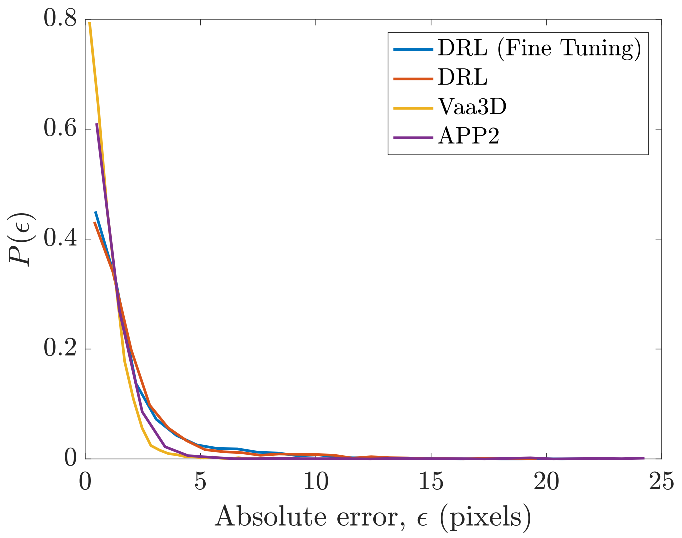


Figure 6: Distribution of errors (p(error) vs. error (px)) accumulated over 3,400 track locations in all 20 images.

Table 5: Coverage (%) and average error ±1 standard deviation (px) of trackers on individual microscopy images.

ID	DRL		DRL (fine-tuned)		APP2		Vaa3D	
	Cov.	Error	Cov.	Error	Cov.	Error	Cov.	Error
1	87.24	1.49 ± 1.60	83.73	1.94 ± 2.50	98.88	1.58 ± 2.10	95.93	1.00 ± 0.91
2	86.01	1.41 ± 1.40	89.66	1.62 ± 1.80	79.68	1.47 ± 1.70	91.97	1.11 ± 1.40
3	94.24	1.90 ± 2.30	94.96	1.78 ± 2.60	99.28	0.93 ± 1.10	98.32	0.79 ± 0.60
4	95.75	1.22 ± 0.94	92.99	1.10 ± 1.30	94.27	0.75 ± 0.96	99.79	0.88 ± 1.20
5	94.68	1.43 ± 1.10	85.69	2.58 ± 2.60	91.41	1.50 ± 2.30	95.71	0.72 ± 0.80
6	78.16	3.78 ± 3.50	92.29	1.78 ± 2.00	92.51	1.47 ± 2.60	91.22	0.90 ± 0.61
7	92.27	1.75 ± 1.20	92.05	1.45 ± 1.40	97.73	1.15 ± 1.30	95.91	0.89 ± 0.79
8	75.72	2.06 ± 1.60	83.39	2.06 ± 1.60	81.47	1.13 ± 1.00	84.35	1.11 ± 0.80
9	49.26	3.79 ± 4.20	92.65	3.50 ± 3.10	35.54	3.98 ± 6.00	94.61	1.49 ± 1.10
10	94.18	1.50 ± 0.49	94.63	1.36 ± 0.41	56.82	2.13 ± 2.1	97.54	1.05 ± 0.77
11	92.22	4.49 ± 4.50	95.56	2.36 ± 2.60	92.96	2.10 ± 3.80	98.52	0.73 ± 0.48
12	72.43	2.06 ± 2.60	76.47	1.46 ± 1.40	79.82	1.13 ± 1.70	90.54	1.01 ± 0.82
13	94.40	1.24 ± 0.96	96.86	1.37 ± 1.50	85.93	3.20 ± 4.70	96.72	0.82 ± 0.68
14	88.73	3.06 ± 2.70	95.77	3.69 ± 3.30	100.00	1.22 ± 1.30	95.21	1.06 ± 0.88
15	88.12	1.41 ± 1.40	86.42	1.97 ± 2.40	73.92	1.76 ± 2.70	68.06	0.74 ± 0.57
16	96.95	1.13 ± 1.10	97.97	1.25 ± 2.60	98.99	0.61 ± 0.45	99.32	0.68 ± 0.55
17	70.14	1.59 ± 1.80	73.44	1.58 ± 1.60	86.06	2.54 ± 4.80	89.46	0.81 ± 0.67
18	66.18	2.10 ± 2.60	70.70	1.57 ± 1.80	48.77	0.62 ± 0.57	67.94	0.64 ± 0.50
19	98.38	0.82 ± 0.75	99.08	1.05 ± 0.79	99.31	0.60 ± 0.58	99.77	0.55 ± 0.42
20	66.85	1.64 ± 1.90	87.23	1.93 ± 2.10	43.00	0.56 ± 0.53	94.40	0.71 ± 0.64
Total	84.10	1.88 ± 2.23	89.08	1.82 ± 2.13	81.82	1.61 ± 2.82	92.26	0.89 ± 0.84

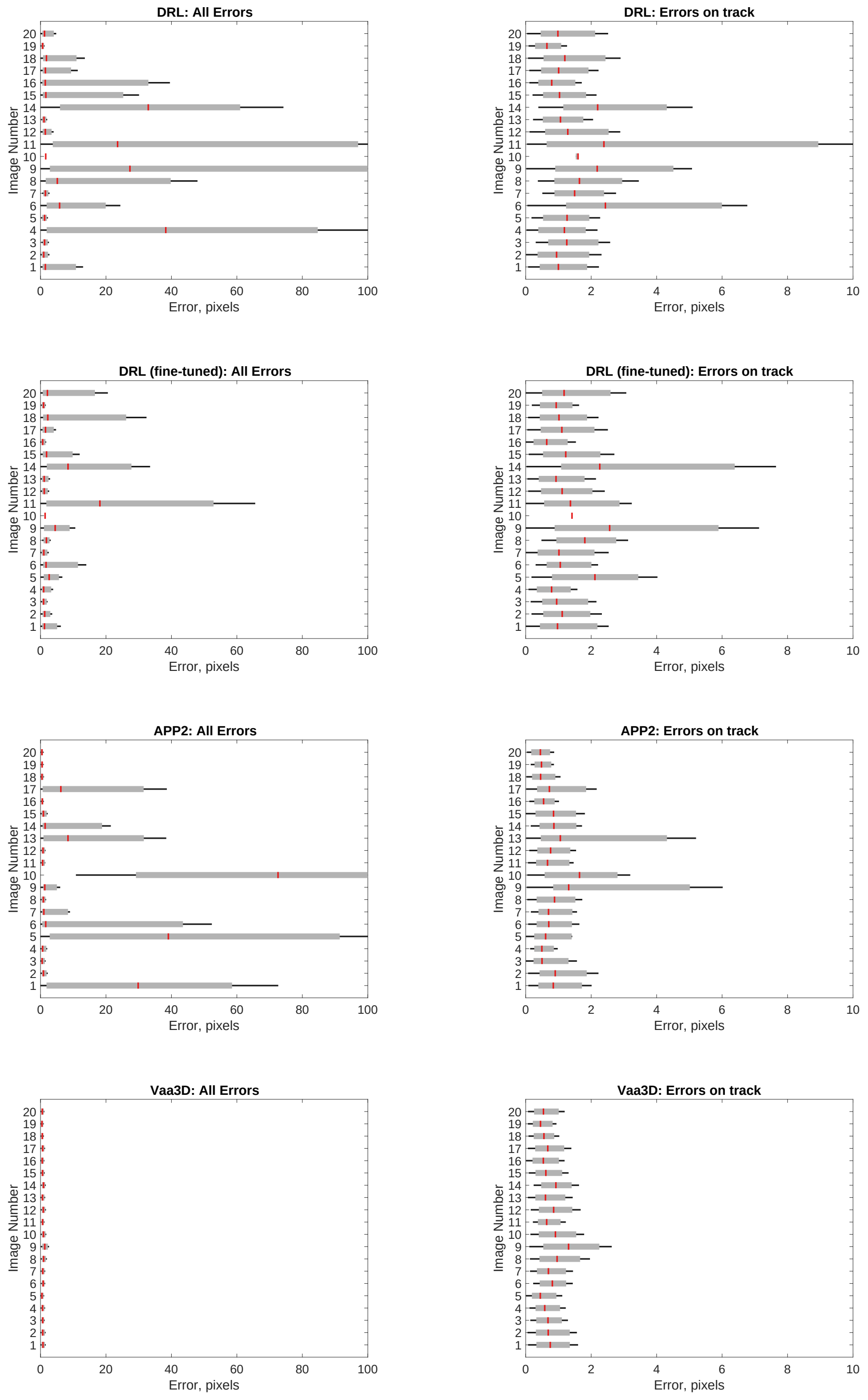


Figure 7: Error distribution (px) of trackers on individual microscopy images, without filtering criterion (left; error range of 0-100px) and with (right; error range of 0-10px). Whiskers set to 0.25× inter-quartile range, for purposes of visibility and comparison.

Table 6: Average entropy ± 1 standard deviation within and outside the threshold ($<$ 3px/$\geq$ 3px). The entropy significantly increases as the tracker passes the threshold for both the DRL tracker ($p < 0.00001$) and the DRL tracker after fine-tuning (FT) ($p < 0.00001$). p-values were calculated using a two-tailed paired t-test on all points within all images.

	DRL		**DRL (FT)**	
ID	Within	Outside	Within	Outside
1	-4.63 ± 0.83	-3.59 ± 0.99	-3.39 ± 0.67	-2.27 ± 1.13
2	-4.17 ± 0.79	-3.43 ± 0.92	-2.83 ± 0.56	-2.06 ± 0.84
3	-4.16 ± 0.64	-3.89 ± 0.65	-2.97 ± 0.55	-2.97 ± 0.62
4	-4.65 ± 0.93	-2.41 ± 0.80	-2.79 ± 0.62	-1.08 ± 0.91
5	-3.96 ± 0.87	-4.09 ± 1.11	-2.63 ± 0.77	-1.21 ± 1.09
6	-3.43 ± 0.93	-2.31 ± 0.70	-2.13 ± 0.68	-0.95 ± 0.58
7	-4.34 ± 0.82	-3.75 ± 0.84	-3.18 ± 0.60	-2.84 ± 0.51
8	-3.86 ± 0.83	-2.60 ± 0.80	-2.75 ± 0.51	-2.52 ± 0.50
9	-2.85 ± 0.75	-2.30 ± 0.85	-2.36 ± 0.60	-1.77 ± 0.84
10	-3.60 ± 0.32	-4.63 ± 1.30	-3.35 ± 0.28	-3.80 ± 0.0
11	-4.23 ± 0.90	-2.58 ± 0.71	-3.34 ± 0.63	-2.03 ± 0.69
12	-3.65 ± 0.87	-2.32 ± 0.70	-2.94 ± 0.62	-2.35 ± 0.46
13	-4.21 ± 0.94	-3.84 ± 0.88	-3.41 ± 0.56	-2.75 ± 0.58
14	-4.26 ± 0.82	-2.99 ± 0.66	-3.43 ± 0.53	-2.59 ± 0.74
15	-4.51 ± 0.79	-2.95 ± 0.89	-3.55 ± 0.45	-2.77 ± 0.55
16	-3.28 ± 1.02	-2.52 ± 0.54	-2.63 ± 0.53	-1.97 ± 0.68
17	-4.16 ± 0.88	-2.25 ± 0.72	-3.23 ± 0.55	-2.21 ± 0.78
18	-3.58 ± 1.20	-2.25 ± 0.67	-2.75 ± 0.62	-1.38 ± 0.75
19	-4.77 ± 0.68	-4.54 ± 1.17	-3.33 ± 0.44	-2.91 ± 0.24
20	-3.54 ± 1.30	-2.39 ± 0.84	-2.75 ± 0.78	-1.85 ± 0.73
Total	-3.99 ± 0.49	-3.08 ± 0.79	-2.99 ± 0.38	-2.21 ± 0.70