This is an Accepted Manuscript version of the following work: László Kopácsi, Árpád Dobolyi, Áron Fóthi, Dávid Keller, Viktor Varga, András Lőrincz: RATS: Robust Automated Tracking and Segmentation of Similar Instances. Artificial Neural Networks and Machine Learning – ICANN 2021, Springer International Publishing 2021. This version of the manuscript has been accepted for publication, after final editorial and peer review is complete, but is not the Version of Record. The final authenticated version is available online at: https://doi.org/10.1007/978-3-030-86365-4_41

RATS: Robust Automated Tracking and Segmentation of Similar Instances*

László Kopácsi[1], Árpád Dobolyi[2], Áron Fóthi[1], Dávid Keller[3], Viktor Varga[1], and András Lőrincz[1]

[1] Department of Artificial Intelligence, Eötvös Loránd University, Budapest, Hungary
{kopacsi,fa2,vv,lorincz}@inf.elte.hu
[2] Department of Physiology and Neurobiology, Eötvös Loránd University, Budapest, Hungary
dobolyi.arpad@ttk.elte.hu
[3] MTA-ELTE Laboratory of Molecular and Systems Neurobiology, Eötvös Loránd University, Budapest, Hungary
keller.david8@gmail.com

Abstract. Continuous identification of objects with identical appearance is crucial to analyze the behavior of laboratory animals. Most existing methods attempt to avoid this problem by excluding direct social interactions or facilitating it by implants or artificial markers. Unfortunately, these techniques may distort the results as they can affect the behavior of the observed animals. In this paper, we present a simple, deep learning-based approach that can overcome these problems by providing reliable segmentation and tracking of similar instances. Recognition of frames where the system could not reliably locate the objects and mark them suggests human supervision is central to the system since there should be no mistake in instance tracking. Manual annotation of these data improves tracking and decreases annotation needs quickly. The proposed method achieves higher segmentation accuracy and more stable tracking than previous methods despite requiring only a small set of manually annotated data.

Keywords: video instance segmentation · tracking · identical objects · deep neural networks.

* The research has been supported by the European Institute of Innovation and Technology. Á.D., Á.F., D.K. and A.L. were supported by Thematic Excellence Programme 2020 (TKP 2020-IKA-05) of the National Research, Development and Innovation Fund of Hungary. A.L. was supported by the Thematic Excellence Programme (Project no. ED_18-1-2019-0030 titled Application-specific highly reliable IT solutions) by the same Fund. V.V. was supported by the the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program. The authors thank to Robert Bosch, Ltd. Budapest, Hungary for their generous support to the Department of Artificial Intelligence.

1 Introduction

High throughput research broke into many parts of neuroscience including e.g. gene expressional studies and in vitro drug discovery and animal studies became a bottleneck to understand neuronal functions. The major output of the nervous system in animals is their behavior, consequently, automated behavioral analysis is a major unsolved problem. The most basic information on the behavior of animals is their location. Tracking of animals has been successful for a large number of small animals when individual identification is not essential [1] or tracking single mice or rats [2]. However, several neuropsychiatric disorders, such as schizophrenia and depression include pathological social interactions while it is the primary problem in others, such as social anxiety disorder and autism spectrum disorder [3]. The study of these disorders as well as understanding the normal brain mechanisms responsible to regulate social interactions require the behavioral analysis of more than one animal. In such cases, genetically modified rats are utilized, which look identical to the human eye, but switching them up can invalidate the findings. Some studies avoid the problem of continuous identification of the animals by excluding direct social interactions [4], but this is not the natural way how animals perform their social behaviors. Therefore, further research is required for automated analysis of social behaviors when direct interactions are allowed even though the continuous identification of similarly looking mice and rats remains a challenge: a common problem in video tracking is the inability to maintain the identity of the animals whenever they are in close proximity. Radio frequency-based methods offer a solution [5, 6], but in this technique transducers have to be implanted into the animals, information other than the position of the animals is hard to obtain, sensors are necessary in the relative proximity of the animals [7], and the need to isolate the animals during their recovery from surgery may have unwanted effects on their behavior. Thus, automated video analysis seems ideal, but the problem of continuous identification of the otherwise similar animals has to be solved. Artificial markers on them, e.g., coloring of the fur may be occluded and may disturb the natural behavior of the animals. Separation of the animals based on the videos only became possible with recent advances in deep learning techniques, which outperformed traditional approaches in tracking single rodents [8, 9]. There have been attempts to use deep learning technologies for quantifying image similarity [10], detecting objects [11], tracking [12,13], and combining velocity of movement with inter-individual distance of rat pairs [14] as well.

In the present study, we developed a combination of algorithms *and* a software package for reliable continuous identification of animals with identical appearance by temporally extending the results of the pre-trained Mask R-CNN [15] model and iteratively fine-tuning on erroneous predictions. Although our solution provides a general framework for neurobiological applications, its parameters are tuned to solve the problem described above. The method begins with synthetic data generation. To this end, we automatically generate overlapping instance annotations with figure-ground segmentation and through various augmentation techniques. After we pre-trained the model, we repeat the follow-

ing steps. (1) We propagate the predictions of the model and mark frames where the system may have made a mistake in accurately locating the instances. (2) Then we check these samples, annotate if necessary and use the new annotations to fine-tune the system. This way, the performance of the overall system can be improved while minimizing the need for hand-annotation *and* our warning system eliminates errors of instance tracking.

To verify that the proposed method can reliably track instances, we count the number of id switches that may have occurred without our warning system. We also measure the instance segmentation capabilities of the system by calculating the mean of region similarity and contour accuracy ($\mathcal{J}\&\mathcal{F}$) measure proposed in the DAVIS benchmark [16]. Due to pre-training, fine-tuning the system on only 30 samples, we found that it can track objects without switching up the instance ids and achieving 0.833 of $\mathcal{J}\&\mathcal{F}$ mean. Therefore, it outperforms previous methods while using a simpler architecture and fewer ground truth samples.

Our contributions are listed below:

- We propose a simple yet efficient method for reliably tracking and segmenting laboratory animals with identical appearance.
- We extended the Tracktor method [12] with instance segmentation.
- We reduce the need to manually annotate a large set of samples by detecting erroneous predictions during tracking.
- Our method achieves state-of-the-art results even though it requires significantly fewer training samples than previous methods.
- The proposed warning method enables high reliability error-free instance tracking from start[4].

2 Methods

Our proposed method consists of several steps (Fig. 1). We start by pre-training Mask R-CNN [15] via synthetic data generation. Then, we extend the model to track objects by propagating the results through time. Meanwhile, we mark samples for annotation by checking the consistency of the predictions between subsequent frames. Lastly, manual annotation and fine-tuning take place. These steps are repeated until the model converges.

2.1 Pre-training

We follow the training method proposed in [17].

Assume a sequence of images $I = [I_1, I_2, \ldots, I_T]$, where $I_t \in [0, 255]^{H \times W \times 3}$ ($t \in [1, T]$). If the camera is stationary and the background is (relatively) constant, we can extract the background $B \in [0, 255]^{H \times W \times 3}$ by taking the most frequently occurring value at each pixel location:

$$B_{h,w,c} = \text{mode}\left(I_{\cdot,h,w,c}\right),$$

[4] Our implementation is available here https://github.com/lkopi/rat_tracking.

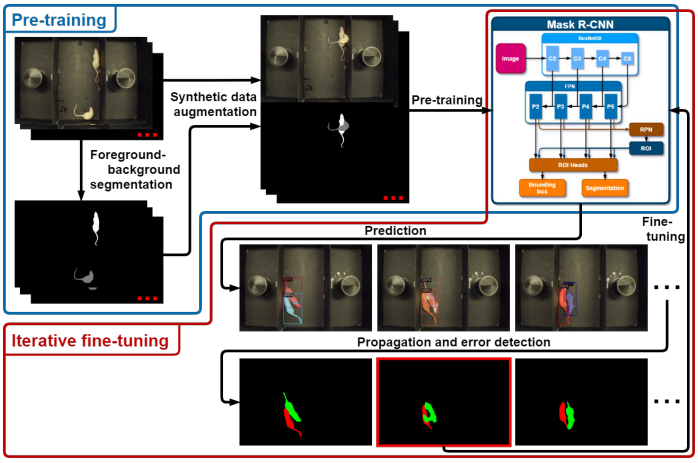

Fig. 1. The pipeline of the proposed method. The method can be divided into two phases. In the first phase, we pre-train the model on the generated synthetic data. While in the second phase, we repeat the following steps until the model converges. We propagate the predictions and detect frames where the model performs poorly. Then use these frames to fine-tune the model. For more information, see Section 2.

where $h \in [1, W], w \in [1, W], c \in [1, 3]$ and mode(.) is the statistical mode operation. Then the foreground-background mask at time step $t \in [1, T]$ can be estimated by

$$M_t = |I_t - B| > \epsilon,$$

where ϵ is an appropriate threshold parameter.

We keep frames where the foreground objects do not occlude each other, then apply various augmentation techniques commonly used in the video segmentation literature [18, 19], such as rotation, scaling, thin-plat-spline warping [20]. This way, we can generate a synthetic dataset and train an instance segmentation method without ground truth data. As our deep neural network architecture, we chose Mask R-CNN [15] with ResNet50 backbone because it is highly modular and can be easily integrated into Tracktor [12].

2.2 Temporal extension using Tracktor

For temporal extension, we used 2 distinct methods. (1) As a baseline, we extended Tracktor [12] to handle instance segmentation. (2) In addition, we propose a simple propagation method described in more details in Section 2.3.

Tracktor is a multi-object tracking method. It predicts the position of the objects by feeding bounding box predictions from the previous frame to the bounding box regression layer of the object detector. Consequently, it accomplishes tracking without requiring further training of the underlying model.

Under the hood, Tracktor is using Faster R-CNN [21] to propagate the bounding box proposals. Mask R-CNN extends Faster R-CNN with an additional seg-

mentation head. Therefore, by replacing Faster R-CNN with Mask R-CNN enables Tracktor to produce instance segmentation as well. This way, we can use our models to track and segment objects.

2.3 Propagation and error detection

We propose a simple algorithm for propagating instance proposals and detecting frames where the model performs poorly.

The method starts with a pre-processing step. Here we remove small regions and apply morphological closing (a dilation followed by erosion) to fill small holes. Then we do various checks to make sure that the predictions are correct:

- We check the number of instances by counting the number of unique ids. We drop the predictions if the number of instances does not equal 2.
- Then, we check whether the sizes of each rat are similar. We estimate the ratio of the area of the rats. If $\frac{\min(area)}{\max(area)} \leq 0.4$, we drop the smaller instance.
- And lastly, we eliminate intertwined predictions. We calculate the overlap between the convex hull of one of the rats and the predicted mask of the other rat. We remove the prediction if the overlap exceeds τ.

After we obtained the filtered result, we propagate instances using bipartite (Hungarian) matching [22]. If we cannot match an instance, we keep the prediction from the preceding frame and mark the frame for further annotation.

Keeping predictions from the previous time step ensures that we do not lose track of objects even when the model could not locate them for several frames. Meanwhile, it provides an adequate segmentation because instances do not change much in such a short time. Moreover, it eliminates the need for re-identification methods, which may struggle when the objects look identical.

2.4 Annotation needs and fine-tuning

By annotating frames marked in the previous step and using them to fine-tune the model, we can increase the accuracy of the system and reduce the id switches during propagation. The annotation need can be mitigated by incrementally tuning the model. During each training cycle, we only provide ground truth for a small subset of samples, then repeat the propagation and error detection step (see Section 2.3) till the model converges. In this phase, we select the frames randomly, and the annotation is done by a human observer, who provides instance segmentation by correcting the predicted masks.

2.5 Benchmark measures

To evaluate our results, we are counting the track switches (TS) and using the region ($\mathcal{J}$) and boundary ($\mathcal{F}$) measure proposed in DAVIS 2017 [16].

Track Switches (TS) measures the number of id switches during tracking. We use it as our primary metric in this paper, as it is crucial to reliably maintain the identity of instances to do further analysis on them.

Let P and $G \in [0, K]^{T \times H \times W}$ be the proposed predictions and ground truth instance masks, respectively. $K = 2$ is the maximum number of instances present on the scene and 0 is kept for the background. We introduce $\tilde{P}_t = [2/P_t]$ which is identical to P_t but the instance ids are switched up.

Now, let $o(X, Y) = \sum_{h \in [1,H] \wedge w \in [1,W]} \mathbf{1}(X_{h,w} = Y_{h,w})$ measures the overlap between X and Y, and $\mathbf{1} : \mathbb{L} \to \{0, 1\}$ is the indicator function defined as

$$\mathbf{1}(l) = \begin{cases} 1, & \text{if } l \text{ is } true \\ 0, & \text{otherwise} \end{cases}.$$

This way, we can calculate the track switches (TS) as in Algorithm 1.

Algorithm 1: Calculation of track switches.

```
Result: TS
TS ← 0;
if o(P_1, G_1) < o(P̃_1, G_1) then
    P, P̃ ← P̃, P;
end
for t = 2, 3 ..., T do
    if o(P_t, G_t) < o(P̃_t, G_t) then
        TS ← TS + 1;
        P, P̃ ← P̃, P;
    end
end
```

Region Similarity ($\mathcal{J}$) is defined as the Intersection over Union (IoU). IoU measures the similarity between the estimated $\mathcal{M}$ and the ground truth $\mathcal{G}$ segmentation. As the name implies it is calculated the following way: $\mathcal{J} = \frac{|\mathcal{M} \cap \mathcal{G}|}{|\mathcal{M} \cup \mathcal{G}|}$.

Contour Accuracy ($\mathcal{F}$) is defined as $\mathcal{F} = \frac{2 P_c R_c}{P_c + R_c}$, where P_c and R_c measures the contour-based precision and recall, respectively between the contour points of the proposed mask $c(\mathcal{M})$ and the ground truth mask $c(\mathcal{G})$. Pixel tolerance is set to the benchmark measure [23].

Region and Boundary ($\mathcal{J}\&\mathcal{F}$) measure is used to evaluate the performance of each method. It is defined as the mean of the region similarity ($\mathcal{J}$) and the contour accuracy ($\mathcal{F}$).

3 Results

3.1 Experimental setup

We used the Mask R-CNN implementation of Detectron2[5] with a ResNet50 backbone pre-trained on the COCO instance segmentation dataset [24].

[5] https://github.com/facebookresearch/detectron2

Table 1. Comparison of results on the test set. Methods where "GT" is ✗ are trained without using any hand-annotated samples, only on synthetic ones generated by the process described in Section 2.1. Tracktor and Proposed used the same weights.

Method	GT	$\mathcal{J}\&\mathcal{F}$ mean ↑	$\mathcal{J}$ mean ↑	$\mathcal{J}$ recall ↑	$\mathcal{J}$ decay ↓	$\mathcal{F}$ mean ↑	$\mathcal{F}$ recall ↑	$\mathcal{F}$ decay ↓	TS ↓
Cluster R-CNN [13]	✗	0.633	0.560	0.645	0.135	0.706	0.763	0.129	45
Tracktor [12]	✗	0.587	0.517	0.584	0.056	0.657	0.759	0.049	174
Proposed	✗	0.787	0.737	0.882	**-0.038**	0.837	0.932	**-0.047**	14
Cluster R-CNN [13]	✓	0.743	0.678	0.790	0.028	0.808	0.813	0.027	23
Tracktor [12]	✓	0.568	0.495	0.575	0.076	0.641	0.722	0.091	112
Proposed	✓	**0.876**	**0.831**	**0.972**	-0.005	**0.922**	**0.987**	-0.004	**0**

For pre-training, we trained the model for 50,000 iterations on an NVIDIA GeForce GTX 1080 graphics card with a batch size of 512 for the heads and a batch size of 4 for the rest of the network. The initial learning rate was set to 0.00025. During the evaluation, the confidence threshold was set to 70%. All other parameters were left as default.

During fine-tuning, we trained the model for 30 epochs on the given subset, and set τ to 1. The rest of the hyper-parameters were left unchanged.

3.2 Database

The database consists of a single video where two identical white rats are interacting in a black container. It was recorded by a stationary camera fixed on the top of the container. The whole video is 58,994 frame-long, and the original resolution was 1280x720, from which we cropped a 640x420 region containing only the box.

For pre-training, we ran the figure-ground segmentation proposed in Section 2.1 on the first 15,000 frames. We kept only frames where instances could be clearly separated, i.e., no occlusion was present. The resulting training set consists of 4,625 images, which we dilated to 27,654 by applying augmentation.

For the test set, we selected 18 200-frame-long clips where the rats are interacting with each other. These samples have been hand-annotated using the VGG Image Annotator [25].

3.3 Temporal extension

We started the experiments by testing the capabilities of each method. We tested each with and without fine-tuning on ground truth samples. We compared the temporal extension techniques described in Section 2.2 and Section 2.3. We pre-trained Mask R-CNN on the training set, then we fine-tuned it on 2,215 hand-annotated samples. We also tested the Cluster R-CNN architecture [13] trained only on synthetic data and after fine-tuning on the ground truth data. In the former case, we used an additional 15,000 frames to pre-train the model as it requires continuous sequences. And in the latter case, it was fine-tuned on 3,000 ground truth samples. The results can be seen on Table 1.

The proposed method performed better on every benchmark measure regardless of the training process. Cluster R-CNN can separate objects even within a

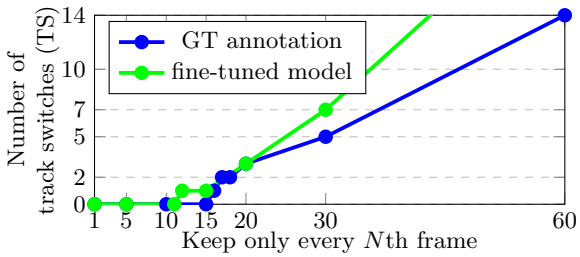

Fig. 2. Robustness of the propagation method. We keep only every Nth ground truth annotation (blue) or prediction (green) and count the track switches. By keeping only every 15th GT or every 11th prediction, the system could maintain stable tracking.

single bounding box, but it suffers from sudden id switches. In some scenarios, the id of the opposite instance may flow through, thereby spoiling the tracklet. Although Tracktor utilizes the same models, its inferior performance was caused by the non-maximum suppression used during tracking as it may fuse detections close to each other, thereby losing track of objects. Meanwhile, in such cases, the proposed method could maintain tracking by propagating previous predictions.

3.4 Robustness

To evaluate the robustness of the proposed temporal extension method, we took the test sequences and kept only every Nth frame. Then propagated the instance segmentation of these frames and counted the track switches (TS). We tested the method using the ground truth annotation and using the best model presented in Section 3.3. Fig. 2 show how the number of TS changes respect to N. As the results show, the system is robust enough to reliably track objects, even if only every 15th ground truth annotation or every 11th prediction was available.

3.5 Error detection and fine-tuning

After we pre-trained the model on the synthetic dataset, we test the performance of the method when it is fine-tuned on samples marked by the system. In every case, we start with pre-trained weights then fine-tune the model for several training cycles. During each training cycle, we randomly select a fixed number of marked samples to tune the model. The results can be seen in Table 2.

As the results show, after training on only 30 samples, the performance of the model increased notably. The $\mathcal{J}\&\mathcal{F}$ mean improved by 6% from 0.787 to 0.833, and the number of track switches reduced to 0. Therefore, the model can learn to track objects with little human effort.

Fig. 3 shows how the number of track switches changes over each iteration. Initially, we started with the pre-trained model, which made 14 TS, then with each iteration, it gradually decreased to 0.

Table 2. Performance of the fine-tuned method. Apart from the "pre-trained" model, "Model" names consist of two numbers: the number of fine-tuning iterations and the number of ground truth samples used during each iteration. The samples were randomly selected from the marked frames.

Model	$\mathcal{J}\&\mathcal{F}$ mean ↑	$\mathcal{J}$ mean ↑	$\mathcal{J}$ recall ↑	$\mathcal{J}$ decay ↓	$\mathcal{F}$ mean ↑	$\mathcal{F}$ recall ↑	$\mathcal{F}$ decay ↓	TS ↓
pre-trained	0.787	0.737	0.882	-0.038	0.837	0.932	-0.047	14
25x1	0.828	0.780	0.947	0.014	0.877	0.979	0.005	0
3x10	0.833	0.785	0.946	0.000	0.881	0.976	-0.010	0
1x100	0.820	0.778	0.935	0.012	0.861	0.977	0.006	0
1x505	0.846	0.797	0.950	0.018	0.895	0.983	0.008	0
1x2215	0.876	0.831	0.972	-0.005	0.922	0.987	-0.004	0

Fig. 4 demonstrates how the length of the marked sequences are distributed. After fine-tuning the model on a few samples, the number of long sequences decreases significantly.

Note that depending on the training method, upon 10-to-15 training samples, the system could reliably track the instances without switches. Hence, there is considerable freedom for the experimenter on risk considerations, checking the potential errors on system warnings, the error types, and the samples that need to be annotated (see, Fig. 2). Optimization of requesting annotations from the experimenter is beyond the present work.

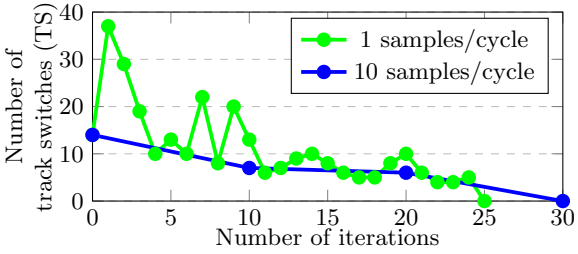

Fig. 3. Performance of the proposed method over the training cycles. "N samples/cycle" denotes that the system was fine-tuned on N randomly selected samples during each training cycle.

Finally, Fig. 5 presents some qualitative results. The proposed method can produce high-quality instance segmentation with minor artifacts. The mask does not capture some fine details, such as the end of the tail and the tip of the nose. Furthermore, when the rats are heavily occluded, the segmentation may extend out to the other mask, but these mistakes vanish when they move away from each other. Nonetheless, the tracking is reliable, and results are adequate to study the animals in an automated way.

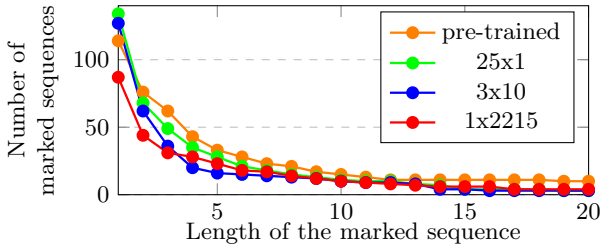

Fig. 4. Distribution of the length of each marked sequence. The plot shows the duration (in frames) of the marked sequences using different models. The notations are the same as in Table 1. Compare with Fig. 2 and see text concerning the role of the experimenter.

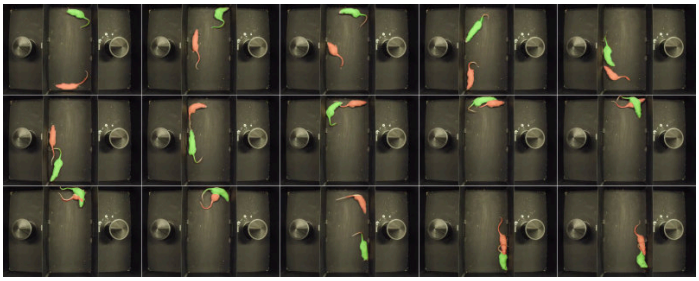

Fig. 5. Qualitative results. Visualization of the propagated predictions on 4 different sequences.

4 Summary

In this paper, we presented a simple and efficient deep learning-based approach to track and segment objects with identical appearance. The proposed method can reliably locate instances with little human effort. We reduced the annotation requirement by marking frames where the system could not provide accurate segmentation. Then we use these frames to fine-tune the model. We repeat these steps until the performance is not sufficient. We evaluated the system concerning its segmentation and tracking accuracy using the region and boundary ($\mathcal{J}\&\mathcal{F}$) measure and track switches, respectively. We showed that it outperforms previous approaches, and it is robust even when the model is unable to locate instances for several frames. Relatively small number of human annotations give rise to (a) switching free tracking of the similar animals in our architecture, while also providing warning signals that can be checked, e.g., by sampling means.

The method becomes suitable for the precise analysis of the social behaviors of rats under the condition that components, such as head, body and tail are recognized. This work is in progress. Our method eliminates the need for direct interactions, introducing implants, or artificial markers. The system could also be applied on other domains with slight modifications.

Although the proposed method performed better than previous ones, the combination of the different approaches could improve the overall performance. Moreover, the utilization of reinforcement learning to optimize the pipeline should also be considered. We note that Mask R-CNN architecture has some shortcomings that could be improved for better localization. Finally, other strategies for propagating masks could increase the robustness, such as using optical flow.

Author Contributions

L.K. and A.L. conceived and designed the research, L.K. and V.V. designed and combined augmentation procedures, L.K. and Á.F. performed computational analyses, Á.D. and D.K. provided the dataset.

References

1. J. Henry and D. Wlodkowic, "High-throughput animal tracking in chemobehavioral phenotyping: current limitations and future perspectives," *Behavioural Processes*, p. 104226, 2020.
2. L. Catarinucci, R. Colella, L. Mainetti, L. Patrono, S. Pieretti, A. Secco, and I. Sergi, "An animal tracking system for behavior analysis using radio frequency identification," *Lab Animal*, vol. 43, no. 9, pp. 321–327, 2014.
3. S. Porcelli, S. Kasper, J. Zohar, D. Souery, S. Montgomery, P. Ferentinos, D. Rujescu, J. Mendlewicz, E. M. Pich, S. Pollentier, *et al.*, "Social dysfunction in mood disorders and schizophrenia: Clinical modulators in four independent samples," *Progress in Neuro-Psychopharmacology and Biological Psychiatry*, vol. 99, p. 109835, 2020.
4. I. Toth and I. D. Neumann, "Animal models of social avoidance and social fear," *Cell and Tissue Research*, vol. 354, no. 1, pp. 107–118, 2013.
5. F. de Chaumont, E. Ey, N. Torquet, T. Lagache, S. Dallongeville, A. Imbert, T. Legou, A.-M. Le Sourd, P. Faure, T. Bourgeron, *et al.*, "Real-time analysis of the behaviour of groups of mice via a depth-sensing camera and machine learning," *Nature Biomedical Engineering*, vol. 3, no. 11, pp. 930–942, 2019.
6. E. J. Mitchell, R. R. Brett, J. D. Armstrong, R. R. Sillito, and J. A. Pratt, "Temporal dissociation of phencyclidine: Induced locomotor and social alterations in rats using an automated homecage monitoring system–implications for the 3rs and preclinical drug discovery," *Journal of Psychopharmacology*, pp. 709–715, 2020.
7. T. Peleh, X. Bai, M. J. Kas, and B. Hengerer, "Rfid-supported video tracking for automated analysis of social behaviour in groups of mice," *Journal of Neuroscience Methods*, vol. 325, p. 108323, 2019.
8. O. Sturman, L. von Ziegler, C. Schl"appi, F. Akyol, M. Privitera, D. Slominski, C. Grimm, L. Thieren, V. Zerbi, B. Grewe, *et al.*, "Deep learning-based behavioral analysis reaches human accuracy and is capable of outperforming commercial solutions," *Neuropsychopharmacology*, vol. 45, no. 11, pp. 1942–1952, 2020.
9. A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "Deeplabcut: markerless pose estimation of user-defined body parts with deep learning," *Nature Neuroscience*, pp. 1281—1289, 2018.

10. G. Charpiat, N. Girard, L. Felardos, and Y. Tarabalka, "Input similarity from the neural network perspective," in *Advances in Neural Information Processing Systems*, vol. 32, pp. 1–10, Curran Associates, Inc., 2019.
11. D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Noscope: Optimizing neural network queries over video at scale," *Proc. VLDB Endow.*, vol. 10, no. 11, p. 1586, 2017.
12. P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *Proceedings of the IEEE international conference on computer vision*, pp. 941–951, 2019.
13. Á. Fóthi, K. B. Faragó, L. Kopácsi, Z. Á. Milacski, V. Varga, and A. Lőrincz, "Multi object tracking for similar instances: A hybrid architecture," in *International Conference on Neural Information Processing*, pp. 436–447, Springer, 2020.
14. S. M. Peters, I. J. Pinter, H. H. Pothuizen, R. C. de Heer, J. E. van der Harst, and B. M. Spruijt, "Novel approach to automatically classify rat social behavior using a video tracking system," *Journal of Neuroscience Methods*, vol. 268, pp. 163–170, 2016.
15. K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.
16. J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, "The 2017 davis challenge on video object segmentation," *arXiv preprint arXiv:1704.00675*, pp. 1–6, 2017.
17. L. Kopácsi, "A self-supervised method for bodypart segmentation and keypoint detection of rat images," in *Collection of Abstracts*, pp. 102–103, 13th Joint Conference on Mathematics and Informatics, 2020.
18. F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A. Sorkine-Hornung, "Learning video object segmentation from static images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2663–2672, 2017.
19. A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele, "Lucid data dreaming for video object segmentation," *International Journal of Computer Vision*, vol. 127, no. 9, pp. 1175–1197, 2019.
20. F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 11, no. 6, pp. 567–585, 1989.
21. S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.
22. A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE international conference on image processing (ICIP)*, pp. 3464–3468, IEEE, 2016.
23. F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 724–732, 2016.
24. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.
25. A. Dutta and A. Zisserman, "The VIA annotation software for images, audio and video," in *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, p. 2276, ACM, 2019.