New idtracker.ai: rethinking multi-animal tracking as a representation learning problem to increase accuracy and reduce tracking times

Jordi Torrents[1†], **Tiago Costa**[1†], **Gonzalo G. de Polavieja**[1*]

[1]Champalimaud Research, Champalimaud Center for the Unknown - Lisbon, Portugal

***For correspondence:** gonzalo.polavieja@neuro.fchampalimaud.org

[†]These authors contributed equally to this work

Abstract idTracker and idtracker.ai approach multi-animal tracking from video as an image classification problem. For this classification, both rely on segments of video where all animals are visible to extract images and their identity labels. When these segments are too short, tracking can become slow and inaccurate and, if they are absent, tracking is impossible. Here, we introduce a new idtracker.ai that reframes multi-animal tracking as a representation learning problem rather than a classification task. Specifically, we apply contrastive learning to image pairs that, based on video structure, are known to belong to the same or different identities. This approach maps animal images into a representation space where they cluster by animal identity. As a result, the new idtracker.ai eliminates the need for video segments with all animals visible, is more accurate, and tracks up to 700 times faster.

Video-tracking systems that attempt to follow individuals frame-by-frame can fail during occlusions, resulting in identity swaps that accumulate over time ***Branson et al.*** (***2009***); ***Plum*** (***2024***); ***Chen et al.*** (***2023***); ***Chiara and Kim*** (***2023***); ***Liu et al.*** (***2023***); ***Bernardes et al.*** (***2021***). idTracker ***Pérez-Escudero et al.*** (***2014***) introduced the paradigm of animal tracking by identification from the animal images. This approach, unfeasible for humans, avoids the accumulation of errors by identity swaps during occlusions. Its successor, idtracker.ai ***Romero-Ferrero et al.*** (***2019***), built on this paradigm by incorporating deep learning and achieved accuracies often exceeding 99.9% in videos of up to 100 animals.

Limitation of the original idtracker.ai

The core idea of the original idtracker.ai is to use a segment of the video in which all animals are visible to extract a set of images and identity labels for each individual. These labeled images are used to train a convolutional neural network (CNN) with one class per animal. Once trained, the network assigns identities to other segments in which all animals are visible. Only segments with identity assignments that meet strict quality criteria are retained, and their images and labels are also used for further training of the CNN. This iterative process of training, assigning, and selecting continues until most of the animal images in the video have been assigned to identities. A second algorithm then tracks animals during crossings.

If no segment exists in which all animals are visible, idtracker.ai cannot start. More commonly,

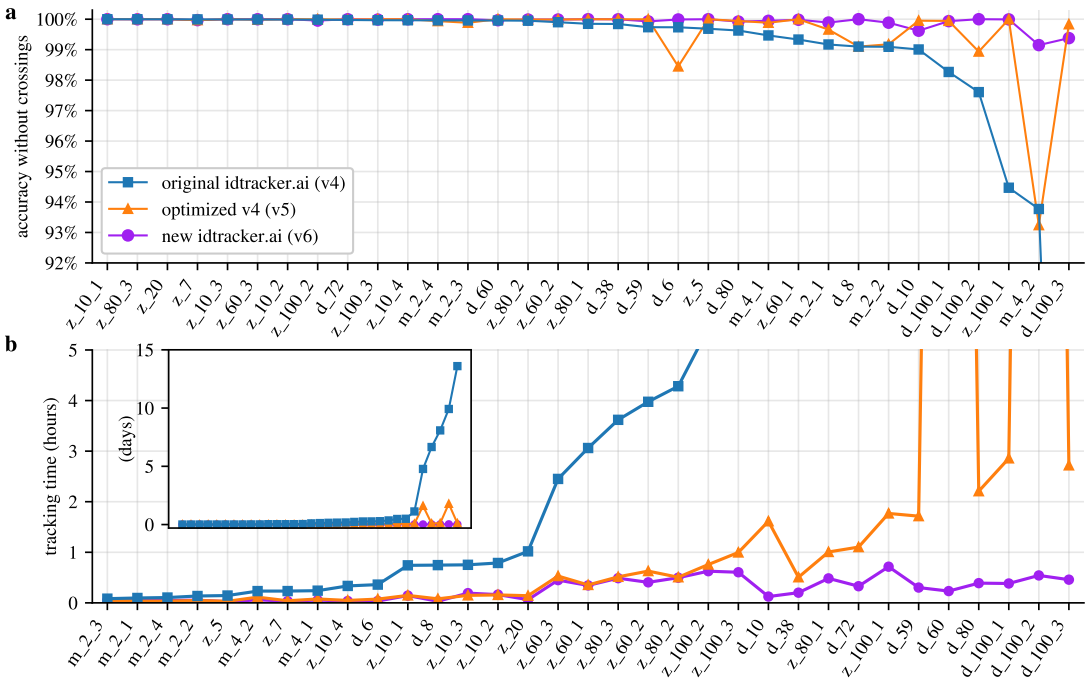


Figure 1. Performance for a benchmark of 33 videos of flies, zebrafish and mice. **a**. Median accuracy was computed using all images of animals in the videos excluding animal crossings. The videos are ordered by decreasing accuracy of the original idtracker.ai results for ease of visualization. **b**. Median tracking times are shown for the scale of hours and, in the inset, for the scale of days. The videos are ordered by increasing tracking times in the original idtracker.ai results for ease of visualization. ***Supplementary Table 1***, ***Supplementary Table 2***, ***Supplementary Table 3*** give more complete statistics (median, mean and 20-80 percentiles) for the original idtracker.ai (version 4 of the software), optimized v4 (version 5) and new idtracker.ai (version 6), respectively. The names of the videos in a. and b. start with a letter for the species (z,d,m), followed by the number of animals in the video, and possibly an extra number to distinguish the video if there are several of the same species and animal group size.

Figure 1—figure supplement 1. Performance for the benchmark with full trajectories with animal crossings

Figure 1—figure supplement 2. Robustness to blurring and light conditions

Figure 1—figure supplement 3. Memory usage across the different softwares

such segments can be short and result in a CNN of low quality. To improve performance in this case, idtracker.ai pretrains the CNN using the entire video, but this process is slow. As a consequence, when the segments in which all animals visible are too short, accuracy might be lower and tracking time longer.

Figure 1a (blue line) gives the accuracies obtained with the original idtracker.ai in our benchmark of 33 videos (see ***Appendix 1*** for details of benchmarking). The first 15 videos of the benchmark are videos of zebrafish, flies (drosophila) and mice with for which of the original idtracker.ai has accuracy of $> 99.9\%$. In the remaining videos the accuracy decreases, reaching 93.77% in video m_4_2, and 69.66% in video d_100_3, which lies outside the plotted range. These accuracies were computed using all animal images in the video while excluding animal crossings. ***Figure 1—figure Supplement 1a*** shows the corresponding results for complete trajectories that include crossings.

Figure 1b (blue line) shows the times that the original idtracker.ai takes to track each of the videos in the benchmark. Some of the videos take a few minutes to track, others a few hours, and six videos take more than three days, one nearly two weeks.

Optimizing idtracker.ai without changes in the learning method

We first optimized idtracker.ai without changing how we identify animals. We improved data loading and redesigned the main objects in the software (see ***Appendix 2*** for details). This version of

the optimized original idtracker.ai (version 5 of the software) achieved higher accuracies, ***Figure 1a*** (orange line), and ***Figure 1—figure Supplement 1a*** (orange line) for results that include animal crossings. The mean accuracy across the benchmark is 99.63% without crossings and 99.49% with crossings, compared with 98.39% without crossings and 98.24% with crossings for the original idtracker.ai.

Tracking times were also significantly reduced. As shown in ***Figure 1b*** (orange line), no video took longer than a day. On average, tracking is 13.5 times faster than with the original version and 120.1 times faster for the more difficult videos. Even so, tracking times of up to one day may still be a limiting factor in some pipelines.

To further improve both accuracy and tracking speed, we retained these optimizations while changing the core identification algorithm to eliminate the need for segments of video in which all animals are visible.

The new idtracker.ai uses representation learning

We reformulate multi-animal tracking as a representation learning problem. In representation learning, we learn a transformation of the input data that makes it easier to perform downstream tasks ***Xing et al.*** (***2002***); ***Bengio et al.*** (***2013***); ***Ericsson et al.*** (***2022***). In our case, the downstream task is to cluster images into animal identities without requiring identity labels.

This reformulation is made possible by the inherent structure of the video, illustrated in ***Figure 2a***. First, we detect specific moments in the video when animals touch or cross paths. These are shown in ***Figure 2a*** as boxes with dashed borders that contain images of overlapping animals. We can then divide the rest of the video into individual fragments, each consisting of the set of images of a single individual between two animal crossings. ***Figure 2a*** shows 14 such fragments as rectangles with a gray background. In addition, a video with N animals may contain global fragments, that is, a collections of N individual fragments that coexist in one or more consecutive frames. An example is shown in ***Figure 2a*** by the five fragments with blue borders. These global fragments are used by the original idtracker.ai to train a CNN. In the new approach, we do not assume that global fragments exist.

Representation learning becomes feasible in this context because we can obtain both positive and negative image pairs without using identity labels. Positive pairs are images of the same individual obtained from within the same individual fragment (***Figure 2a***, green boxes). Negative pairs are images of different individuals taken from different individual fragments that coexist in time for one or more frames (***Figure 2a***, red boxes). We can then use the positive and negative pairs of images for contrastive learning, a self-supervised learning framework designed to learn a representation space in which positive examples are close together, and negative examples are far apart ***Schroff et al.*** (***2015***); ***Dong and Shen*** (***2018***); ***KAYA and BİLGE*** (***2019***); ***Chen et al.*** (***2020a***,b); ***Guo et al.*** (***2020***); ***Wang et al.*** (***2020***); ***Yang et al.*** (***2020***). This formulation is supported by the success of deep metric and contrastive learning methods (see ***Appendix 3*** for comparison with previous work).

We first evaluated which neural networks are suitable for contrastive learning with animal images. In addition to our previous CNN from idtracker.ai, we tested 31 networks from 10 different families of state-of-the-art convolutional neural networks and transformer architectures, selected for their compatibility with consumer-grade GPUs and ability to handle small input images (20×20 to 100×100 pixels) typical in collective animal behavior videos. Among these architectures, ResNet18 (v1) ***He et al.*** (***2016a***) without pretrained weights was the fastest to obtain low errors (see ***Appendix 3***).

A ResNet18 with M outputs maps each input image to a point in an M-dimensional representation space (illustrated in ***Figure 2b*** as a point on a plane). Experiments showed that using $M = 8$ achieved faster convergence to low error (see ***Appendix 3***). ResNet18 is trained using a contrastive loss function (***Chopra et al.*** (***2005***), see ***Appendix 3*** for details). Each image in a positive or negative pair is input separately into the network, producing a point in the 8-dimensional representation

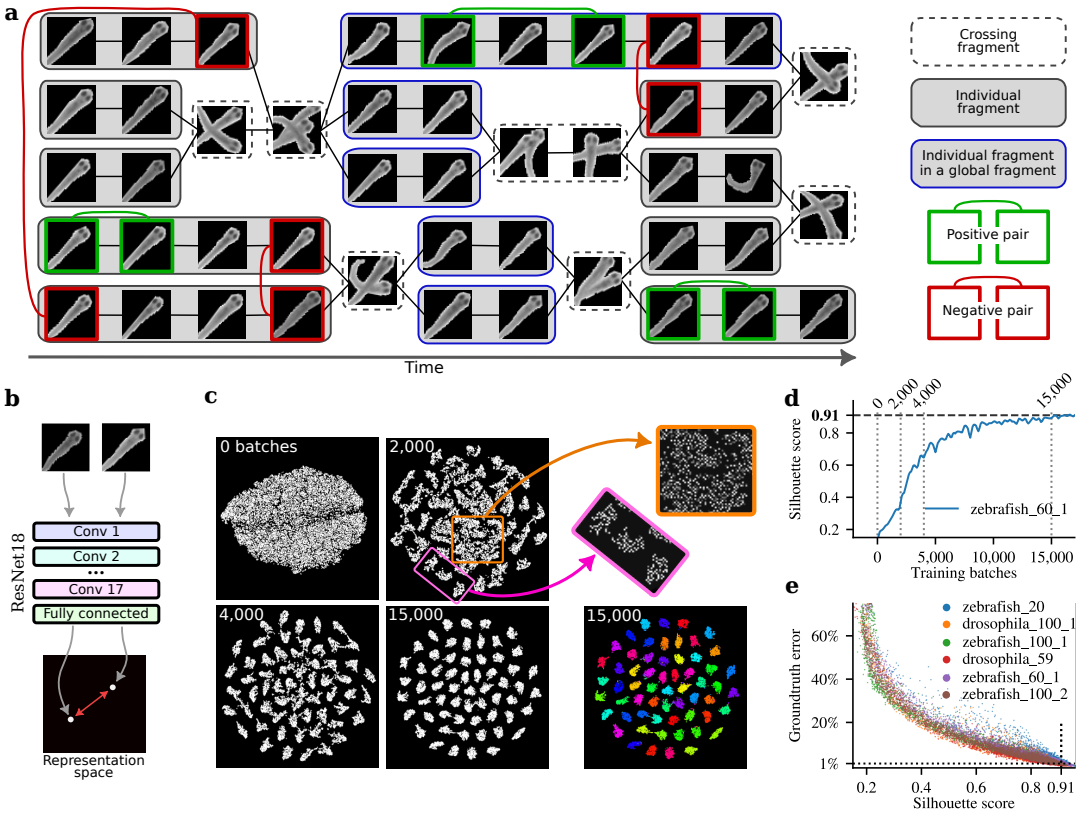


Figure 2. Tracking by identification using deep contrastive learning. a Schematic representation of a video with five fish. **b** A ResNet18 network with 8 outputs generates a representation of each animal image as a point in an 8-dimensional space (here shown in 2D for visualization). Each pair of images corresponds to two points in this space, separated by a Euclidean distance. The ResNet18 network is trained to minimize this distance for positive pairs and maximize it for negative pairs. **c** 2D t-SNE visualizations of the learned 8-dimensional representation space. Each dot represents an image of an animal from the video. As training progresses, clusters corresponding to individual animals become clearer, plotted at training with 0, 2,000, 4,000 and 15,000 batches (each batch contains 400 positive and 400 negative pairs of images). The t-SNE plot at 15,000 training batches is also shown color-coded by human-validated ground-truth identities. The pink rectangle at 2,000 batches of training highlights clear clusters and the orange square fuzzy clusters. **d** The Silhouette score measures cluster coherence and increases during training, as illustrated for a video with 60 zebrafish. **e** A Silhouette score of 0.91 corresponds to a human-validated error rate of less than 1% per image.

space. For an image pair, we then obtain two points in an 8-dimensional space, separated by some (Euclidean) distance. The optimization of the loss function minimizes (or maximizes) this Euclidean distance for positive (or negative) pairs until the distance D_{pos} (or D_{neg}) is reached. The effect of D_{pos} is to prevent the collapse to a single of the positive images coming from the same fragment, allowing for a small region of the 8-dimensional representation space for all the positive pairs of the same identity. The effect of D_{neg} is to prevent excessive scatter of the points representing images from negative pairs. We empirically determined that $D_{\text{neg}}/D_{\text{pos}} = 10$ results in a faster method to obtain low error (see ***Appendix 3***), and we use $D_{\text{pos}} = 1$ and $D_{\text{neg}} = 10$.

As the model trains, the representation space becomes increasingly structured, with similar data points forming coherent clusters. ***Figure 2c*** visualizes this progression using 2D t-SNE ***van der Maaten and Hinton*** (***2008***) plots of the 8-dimensional representation space. After $2,000$ training batches, initial clusters emerge, and by 15,000 batches, distinct clusters corresponding to individual animals are evident. Ground truth identities verified by humans confirm that each cluster corresponds to an animal identity (***Figure 2c***, colored clusters).

The method to select positive and negative pairs is critical for fast learning ***Awasthi et al.*** (***2022***);

Khosla et al. (*2021*); *Rösch et al.* (*2024*). This is because not all image pairs contribute equally to training. *Figure 2c* shows at $2,000$ training batches that some clusters are well-defined (e.g. those inside the pink rectangle) while others remain fuzzy (e.g. those inside the orange square). Images in well-defined clusters have negligible impact on the loss or weight updates, as positive pairs are already close and negative pairs are sufficiently separated. Sampling from these well-defined clusters, therefore, wastes time. In contrast, fuzzy clusters contain images that still contribute significantly to the loss and benefit from further training. To address this, we developed a sampling method that prioritizes pairs from underperforming clusters requiring additional learning, while maintaining baseline sampling for all clusters based on fragment size (see *Appendix 3*). This ensures consistent updates across the representation space and prevents forgetting in well-defined clusters.

To assign identities to animal images, we perform K-means clustering *Sculley* (*2010*) on the points representing all images of the video in the learned 8-dimensional representation space. Each image is then assigned to a cluster with a probability that increases the closer it is to the cluster center. To evaluate clustering quality, we compute the mean Silhouette score *Rousseeuw* (*1987*), which quantifies intra-cluster cohesion and inter-cluster separation. A maximum value of 1 indicates ideal clustering. During training, the mean Silhouette score increases (*Figure 2d*). We empirically determined that a value of 0.91 for this index corresponds to an identity assignment error below 1% for a single image (*Figure 2e*). As a result, we use 0.91 as the stopping criterion for training (see *Appendix 3*).

After the assignment of identities to images of animals, we run some steps that are common to the previous idtracker.ai. For example, we make a final assignment of all images in fragments as each fragment must have all assignments to be the same, eliminating some errors in individual images. Also, an algorithm already present in idTracker assigns identities in the animal's crossings taking into account that we know the identities before and after.

The new idtracker.ai (v6) has a higher accuracy than original idtracker.ai (v4) and than its optimized version (v5), *Figure 1a* (purple line). Its average accuracy in the benchmark is 99.92% and 99.82% without and with crossings, respectively, an important improvement over the original idtracker.ai v4 (98.39% and 98.24%) and its optimized version v5 (99.63% and 99.49%). It also gives much shorter times than the original idtracker.ai (v4) and its optimized version (v5), *Figure 1b* (purple line). It is on average 90 times faster than the original idtracker.ai (v4) and, for the more difficult videos, up to 712 times faster.

As for the original idtracker.ai, the new idtracker.ai can work well with lower resolutions, blur and video compression, and with inhomogeneous light (*Figure 1—figure Supplement 2*). We also compared the new idtracker.ai to TRex *Walter and Couzin* (*2021*), which is based on idtracker.ai without pretraining and with additional operations like eroding crossings to make global fragments longer, posture image normalization, tracklet subsampling, and the use of uniqueness feedback during training. TRex gives comparable accuracies to the original idtracker.ai in the benchmark, and it is on average 31 times faster than the original idtracker.ai and up to 316 times faster (*Figure 1—figure Supplement 1b*). However, the new idtracker.ai is both more accurate and faster than TRex (*Figure 1—figure Supplement 1*). The mean accuracy of TRex across the benchmark is 98.14% and 97.89% excluding and including animal crossings, respectively. This is noticeably below the values for the new idtracker.ai of 99.92% and 99.82%, respectively. Also, the new idtracker.ai is on average 4.5 times faster and up to 16.5 times faster than TRex. Additionally, the new idtracker.ai has a memory peak lower than TRex (*Figure 1—figure Supplement 3*).

No need for global fragments

The new idtracker.ai also works in videos in which the original idtracker.ai does not even track because there are no global fragments. Global fragments are absent in videos with very extensive animal occlusions, for example because animals touch or cross more frequently, parts of the setup are covered, or the camera focuses on only a specific region of the setup.

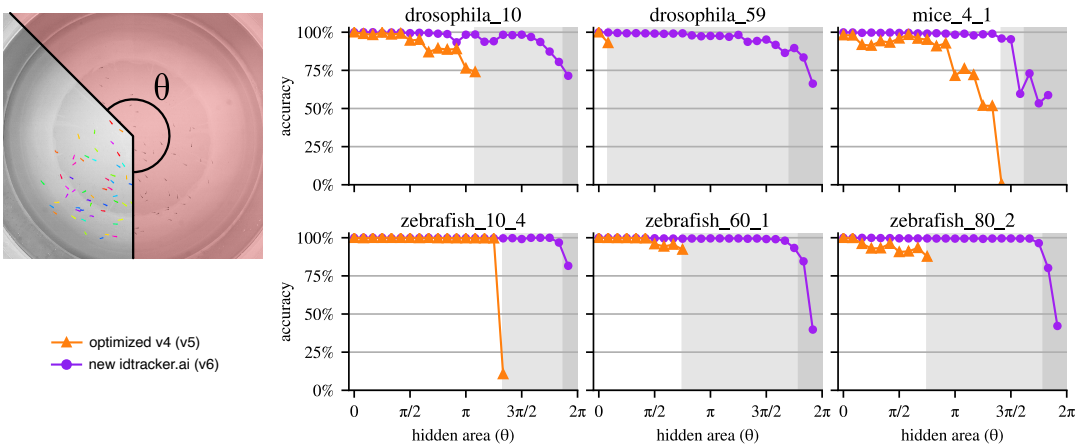


Figure 3. Tracking with strong occlusions. Accuracies when we mask a region of a video defined by an angle θ and the tracking system has no access to the information behind the mask. Light and dark gray region correspond to the angles for which no global fragments exist in the video. Dark gray regions correspond to angles for which the video has a fragment connectivity lower than 0.5, with the fragment connectivity defined as the average number of other fragments each fragment coexists with, divided by $N-1$, with N the total number of animals; see ***Figure 3—figure Supplement 1***, for an analysis justifying this value of 0.5 . The original idtracker.ai (v4) and its optimized version (v5) cannot work in the gray regions, and new idtracker.ai is expected to deteriorate only in the dark gray region.

Figure 3—figure supplement 1. Fragment connectivity analysis

To systematically evaluate this, we digitally masked videos with a sector of angle θ (***Figure 3***). Masked pixels were enforced as background during segmentation, causing animals to disappear from the video when entering the occluded region. Consequently, each re-entry into the visible area requires re-identification with no knowledge of the identities in the occluded region (see ***Methods*** for more details about the occlusion tests).

The light and dark gray regions in ***Figure 3*** correspond to videos with no global fragments, and the original idtracker.ai and its optimized version declare tracking impossible in these regions. The new idtracker.ai, however, works well until approximately $1/4$ of the setup is visible, and afterward it degrades. This shows the limit of the new idtracker.ai. The deterioration happens because, for the clustering process to be successful, we need enough coexisting individual fragments to have both positive and negative examples. We measure this using fragment connectivity, defined as the average number of other fragments a fragment coexists with, divided by $N-1$, with N the total number of animals in the video. Empirically, we find that a fragment connectivity below 0.5 corresponds to low accuracies ***Figure 3—figure Supplement 1***. The new idtracker.ai flags when this condition of low fragment connectivity takes place, which we indicate in ***Figure 3*** with the dark gray regions.

Output of new idtracker.ai

The final output of the new idtracker.ai consists of the $x-y$ coordinates for each identified animal and video frame. Additionally, it provides an estimation of the achieved accuracy, the Silhouette score as a measure of clustering quality, and the probability of correct identity assignment for each animal and frame. The new idtracker.ai also includes the following tools, for which we also give an example workflow in ***Appendix 4*** and documentation at https://idtracker.ai/latest/user_guide/tools.html:

- `idtrackerai_inspect_clusters`, a tool to visually inspect the learned representation space and check the clusters integrity.
- `idtrackerai_validate`, a graphic app to review and correct tracking results.
- `idtrackerai_video`, a graphic app to generate videos of the computed animal trajectories

on top of the original video for visualization. This app also generates individual videos for each animal showing only its cropped region over time to be able to run pose estimators like the ones in ***Lauer et al.*** (***2022***); ***Pereira et al.*** (***2022***); ***Segalin et al.*** (***2021***); ***Tang et al.*** (***2025***); ***Biderman et al.*** (***2024***).
- `idmatcher.ai`, a tool to match identities across multiple recordings (see ***Appendix 5***).
- Direct integration with SocialNet, a model of collective behavior introduced in ***Heras et al.*** (***2019***).
- Direct integration with *trajectorytools*, a Python package for 2D trajectory processing, and a set of Jupyter Notebooks that uses *trajectorytools* to analyze basic movement properties and spatial relationships between the animals.

Acknowledgments

We thank Alfonso Perez-Escudero, Paco Romero-Ferrero, Francisco J. Hernandez Heras, Madalena Valente for discussions, and three anonymous reviewers for many suggestions. This work was supported by Fundaçao para a Ciência e Tecnologia PTDC/BIA-COM/5770/2020 (to G.G.dP.) and Champalimaud Foundation (to G.G.dP.).

Author contributions

T.C. and G.G.dP. devised project and main algorithm, T.C. performed tests of the algorithm as stand alone, J.T. developed version 5, implemented the new algorithm into idtracker.ai architecture and made final tests with help from T.C., G.G.dP. supervised project, J.T. and T.C. wrote the Methods and Appendices with help from G.G.dP., and G.G.dP. wrote the main text with help from J.T and T.C.

Methods

Software availability

idtracker.ai is a free and open source project (license GPLv3). Information about its installation and usage can be found on the website https://idtracker.ai/. The source code is available in https://gitlab.com/polavieja_lab/idtrackerai and the package is pip-installable from PyPI. All versions can be found in these platforms, specifically *"original idtracker.ai (v4)"* as v4.0.12, *"optimized v4 (v5)"* as v5.2.12 and *"new idtracker.ai (v6)"* as v6.0.9. We only actively maintain and provide support for the latest version available, having the old ones for archive and reference only.

Data availability

All videos used in this study, their tracking parameters and human-validated groundtruth can be found in our data repository at https://idtracker.ai/.

Tested computer specifications

The software idtracker.ai depends on PyTorch and is thus compatible with any machine that can run PyTorch, including Windows, MacOS, and Linux systems. Although no specific hardware is required, a graphics card is highly recommended for hardware-accelerated machine-learning computations.

Version 6 of idtracker.ai was tested on computers running Ubuntu 24.04, Fedora 41, Windows 11 with NVIDIA GPUs from the 1000 to the 4000 series, and MacOS 15 with Metal chips. The benchmark results presented in this study were performed on a desktop computer running Ubuntu 24.04 LTS 64bit with a AMD Ryzen 9 5950X (32 cores at 3.4 GHz) processor, 128 GB RAM and an NVIDIA GeForce RTX 4090.

Occlusion tests

We clarify here some details from the occlusion tests presented in the section ***No need for global fragments***. The occlusion tests were performed using the same set of videos as in the benchmark. For these tests, we defined an occlusion mask as a region of interest in the software, applied during the segmentation step. When video frames were converted into binary foreground-background images, all pixels inside the mask were treated as background.

The fragments are detected as in any other video. Specifically, this means that when one animal is hidden in the mask, the animal is lost and the fragment breaks. Fragments are not built adding artificial links.

For evaluation, the ground truth contained the positions of all animals at all times, but, obviously, only positions outside the mask were used to compute tracking accuracy. To avoid partial detections of the animals, trajectories within 15 pixels of the mask boundary (75 pixels for mice videos) were excluded from the evaluation.

References

Awasthi P, Dikkala N, Kamath P, Do More Negative Samples Necessarily Hurt in Contrastive Learning?; 2022. https://arxiv.org/abs/2205.01789.

Bengio Y, Courville A, Vincent P. Representation Learning: A Review and New Perspectives. IEEE Trans Pattern Anal Mach Intell. 2013 Aug; 35(8):1798–1828. https://doi.org/10.1109/TPAMI.2013.50, doi: 10.1109/TPAMI.2013.50.

Bernardes RC, Lima MAP, Guedes RNC, da Silva CB, Martins GF. Ethoflow: Computer Vision and Artificial Intelligence-Based Software for Automatic Behavior Analysis. Sensors. 2021; 21(9). https://www.mdpi.com/1424-8220/21/9/3237, doi: 10.3390/s21093237.

Bernardin K, Stiefelhagen R. Evaluating multiple object tracking performance: the CLEAR MOT metrics. EURASIP Journal on Image and Video Processing. 2008; 2008(1):246309.

Biderman D, Whiteway MR, Hurwitz C, Greenspan N, Lee RS, Vishnubhotla A, Warren R, Pedraja F, Noone D, Schartner MM, Huntenburg JM, Khanal A, Meijer GT, Noel JP, Pan-Vazquez A, Socha KZ, Urai AE, Cunningham JP, Sawtell NB, Paninski L. Lightning Pose: improved animal pose estimation via semi-supervised learning, Bayesian ensembling and cloud-native open-source tools. Nature Methods. 2024 July; 21(7):1316–1328. doi: 10.1038/s41592-024-02319-1.

Branson K, Robie AA, Bender J, Perona P, Dickinson MH. High-throughput ethomics in large groups of Drosophila. Nature Methods. 2009; 6(6):451–457. https://doi.org/10.1038/nmeth.1328, doi: 10.1038/nmeth.1328.

Chen T, Kornblith S, Norouzi M, Hinton G. A simple framework for contrastive learning of visual representations. In: *Proceedings of the 37th International Conference on Machine Learning* ICML'20, JMLR.org; 2020. p. 11.

Chen T, Kornblith S, Swersky K, Norouzi M, Hinton G. Big self-supervised models are strong semi-supervised learners. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems* NIPS '20, Red Hook, NY, USA: Curran Associates Inc.; 2020. p. 13.

Chen X, He K. Exploring Simple Siamese Representation Learning. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; 2021. p. 15745–15753. doi: 10.1109/CVPR46437.2021.01549.

Chen Z, Zhang R, Fang HS, Zhang YE, Bal A, Zhou H, Rock RR, Padilla-Coreano N, Keyes LR, Zhu H, Li YL, Komiyama T, Tye KM, Lu C. AlphaTracker: a multi-animal tracking and behavioral analysis tool. Frontiers in Behavioral Neuroscience. 2023; 17. https://www.frontiersin.org/journals/behavioral-neuroscience/articles/10.3389/fnbeh.2023.1111908, doi: 10.3389/fnbeh.2023.1111908.

Chiara V, Kim SY. AnimalTA: A highly flexible and easy-to-use program for tracking and analysing animal movement in different environments. Methods in Ecology and Evolution. 2023; 14(7):1699–1707. https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.14115, doi: https://doi.org/10.1111/2041-210X.14115.

Chopra S, Hadsell R, LeCun Y. Learning a similarity metric discriminatively, with application to face verification. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1; 2005. p. 539–546 vol. 1. doi: 10.1109/CVPR.2005.202.

Crouse DF. On implementing 2D rectangular assignment algorithms. IEEE Transactions on Aerospace and Electronic Systems. 2016; 52(4):1679–1696. doi: 10.1109/TAES.2016.140952.

Dong X, Shen J. Triplet Loss in Siamese Network for Object Tracking. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y, editors. *Computer Vision – ECCV 2018* Cham: Springer International Publishing; 2018. p. 472–488.

Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale; 2020.

Ericsson L, Gouk H, Loy CC, Hospedales TM. Self-Supervised Representation Learning: Introduction, advances, and challenges. IEEE Signal Processing Magazine. 2022; 39(3):42–62. doi: 10.1109/MSP.2021.3134634.

Grill J, Strub F, Altché F, Tallec C, Richemond PH, Buchatskaya E, Doersch C, Avila Pires B, Guo ZD, Gheshlaghi Azar M, Piot B, Kavukcuoglu K, Munos R, Valko M. Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning. In: *Advances in Neural Information Processing Systems (NeurIPS)*; 2020. .

Grunau C, Özüdoğru AA, Rozhoň V, Tětek J, A Nearly Tight Analysis of Greedy k-means++; 2022.

Guo S, Xu P, Miao Q, Shao G, Chapman CA, Chen X, He G, Fang D, Zhang H, Sun Y, Shi Z, Li B. Automatic Identification of Individual Primates with Deep Learning Techniques. iScience. 2020; 23(8):101412. https://www.sciencedirect.com/science/article/pii/S2589004220306027, doi: https://doi.org/10.1016/j.isci.2020.101412.

He K, Chen X, Xie S, Li Y, Dollár P, Girshick R. Masked Autoencoders Are Scalable Vision Learners. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; 2022. p. 15979–15988. doi: 10.1109/CVPR52688.2022.01553.

He K, Fan H, Wu Y, Xie S, Girshick R. Momentum Contrast for Unsupervised Visual Representation Learning. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; 2020. p. 9726–9735. doi: 10.1109/CVPR42600.2020.00975.

He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 2016. p. 770–778. doi: 10.1109/CVPR.2016.90.

He K, Zhang X, Ren S, Sun J. Identity Mappings in Deep Residual Networks. In: Leibe B, Matas J, Sebe N, Welling M, editors. *Computer Vision – ECCV 2016* Cham: Springer International Publishing; 2016. p. 630–645.

Heras FJH, Romero-Ferrero F, Hinz RC, de Polavieja GG. Deep attention networks reveal the rules of collective motion in zebrafish. PLOS Computational Biology. 2019 09; 15(9):1–23. https://doi.org/10.1371/journal.pcbi.1007354, doi: 10.1371/journal.pcbi.1007354.

Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications; 2017.

Huang G, Liu Z, van der Maaten L, Weinberger KQ, Densely Connected Convolutional Networks; 2016.

Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size; 2016.

KAYA M, BİLGE HS. Deep Metric Learning: A Survey. Symmetry. 2019; 11(9). https://www.mdpi.com/2073-8994/11/9/1066, doi: 10.3390/sym11091066.

Khosla P, Teterwak P, Wang C, Sarna A, Tian Y, Isola P, Maschinot A, Liu C, Krishnan D, Supervised Contrastive Learning; 2021. https://arxiv.org/abs/2004.11362.

Kingma DP, Ba J, Adam: A Method for Stochastic Optimization; 2017. https://arxiv.org/abs/1412.6980.

Krizhevsky A, One weird trick for parallelizing convolutional neural networks; 2014.

Lauer J, Zhou M, Ye S, Menegas W, Schneider S, Nath T, Rahman MM, Santo VD, Soberanes D, Feng G, Murthy VN, Lauder G, Dulac C, Mathis MW, Mathis A. Multi-animal pose estimation, identification and tracking with DeepLabCut. Nature Methods. 2022 April; 19(4):496–504. https://doi.org/10.1038/s41592-022-01443-0, doi: 10.1038/s41592-022-01443-0.

Li W, Zhao R, Xiao T, Wang X. DeepReID: Deep Filter Pairing Neural Network for Person Re-identification. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*; 2014. p. 152–159. doi: 10.1109/CVPR.2014.27.

Liang C, Zhang Z, Lu Y, Zhou X, Li B, Ye X, Zou J. Rethinking the Competition Between Detection and ReID in Multiobject Tracking. IEEE Transactions on Image Processing. 2020; 31:3182–3196. https://api.semanticscholar.org/CorpusID:225062305.

Liu S, Han L, Liu X, Ren J, Wang F, YingLiu, Lin Y, FishMOT: A Simple and Effective Method for Fish Tracking Based on IoU Matching; 2023. https://arxiv.org/abs/2309.02975.

Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B, Swin Transformer: Hierarchical Vision Transformer using Shifted Windows; 2021.

van der Maaten L, Hinton G. Visualizing Data using t-SNE. Journal of Machine Learning Research. 2008; 9(86):2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html.

Oquab M, Darcet T, Moutakanni T, Vo H, Szafraniec M, Khalidov V, Fernandez P, Haziza D, Massa F, El-Nouby A, Assran M, Ballas N, Galuba W, Howes R, Huang PY, Li SW, Misra I, Rabbat M, Sharma V, Synnaeve G, et al. DINOv2: Learning Robust Visual Features without Supervision. . 2024; https://arxiv.org/abs/2304.07193.

Pereira TD, Tabris N, Matsliah A, Turner DM, Li J, Ravindranath S, Papadoyannis ES, Normand E, Deutsch DS, Wang ZY, McKenzie-Smith GC, Mitelut CC, Castro MD, D'Uva J, Kislin M, Sanes DH, Kocher SD, Wang SSH, Falkner AL, Shaevitz JW, et al. SLEAP: A deep learning system for multi-animal pose tracking. Nature Methods. 2022 April; 19(4):486–495. https://doi.org/10.1038/s41592-022-01426-1, doi: 10.1038/s41592-022-01426-1.

Pérez-Escudero A, Vicente-Page J, Hinz RC, Arganda S, De Polavieja GG. idTracker: tracking individuals in a group by automatic identification of unmarked animals. Nature methods. 2014; 11(7):743–748.

Plum F. OmniTrax: A deep learning-driven multi-animal tracking and pose-estimation add-on for Blender. Journal of Open Source Software. 2024; 9(95):5549. https://doi.org/10.21105/joss.05549, doi: 10.21105/joss.05549.

Ristani E, Solera F, Zou R, Cucchiara R, Tomasi C. Performance Measures and a Data Set for Multi-target, Multi-camera Tracking. In: Hua G, Jégou H, editors. *Computer Vision – ECCV 2016 Workshops* Cham: Springer International Publishing; 2016. p. 17–35.

Robinson JD, Chuang CY, Sra S, Jegelka S. Contrastive Learning with Hard Negative Samples. In: *International Conference on Learning Representations*; 2021. https://openreview.net/forum?id=CR1XOQ0UTh-.

Romero-Ferrero F, Bergomi MG, Hinz RC, Heras FJ, De Polavieja GG. Idtracker. ai: tracking all individuals in small or large collectives of unmarked animals. Nature methods. 2019; 16(2):179–182.

Romero-Ferrero F, Heras FJ, Rance D, de Polavieja GG. A study of transfer of information in animal collectives using deep learning tools. Philosophical Transactions of the Royal Society B. 2023; 378(1874):20220073.

Rousseeuw PJ. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics. 1987; 20:53–65. https://www.sciencedirect.com/science/article/pii/0377042787901257, doi: https://doi.org/10.1016/0377-0427(87)90125-7.

Rösch PJ, Oswald N, Geierhos M, Libovický J, Enhancing Conceptual Understanding in Multimodal Contrastive Learning through Hard Negative Samples; 2024. https://arxiv.org/abs/2403.02875.

Schroff F, Kalenichenko D, Philbin J. FaceNet: A unified embedding for face recognition and clustering. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* IEEE; 2015. http://dx.doi.org/10.1109/CVPR.2015.7298682, doi: 10.1109/cvpr.2015.7298682.

Sculley D. Web-scale k-means clustering. In: *Proceedings of the 19th International Conference on World Wide Web* WWW '10, New York, NY, USA: Association for Computing Machinery; 2010. p. 1177–1178. https://doi.org/10.1145/1772690.1772862, doi: 10.1145/1772690.1772862.

Segalin C, Williams J, Karigo T, Hui M, Zelikowsky M, Sun JJ, Perona P, Anderson DJ, Kennedy A. The Mouse Action Recognition System (MARS) software pipeline for automated analysis of social behaviors in mice. eLife. 2021 nov; 10:e63720. https://doi.org/10.7554/eLife.63720, doi: 10.7554/eLife.63720.

Shrivastava A, Gupta A, Girshick R. Training Region-Based Object Detectors with Online Hard Example Mining. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 2016. p. 761–769. doi: 10.1109/CVPR.2016.89.

Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, Le QV, MnasNet: Platform-Aware Neural Architecture Search for Mobile; 2018.

Tang G, Han Y, Sun X, Zhang R, Han M, Liu Q, Wei P. Anti-drift pose tracker (ADPT): A transformer-based network for robust animal pose estimation cross-species. eLife. 2025 Mar; http://dx.doi.org/10.7554/eLife.95709.2, doi: 10.7554/elife.95709.2.

Teh CH, Chin RT. On the detection of dominant points on digital curve. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 1989 09; 11:859 – 872. doi: 10.1109/34.31447.

Walter T, Couzin ID. TRex, a fast multi-animal tracking system with markerless identification, and 2D estimation of posture and visual fields. eLife. 2021 feb; 10:e64000. https://doi.org/10.7554/eLife.64000, doi: 10.7554/eLife.64000.

Wang Z, Zheng L, Liu Y, Li Y, Wang S. Towards Real-Time Multi-Object Tracking. In: *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI* Berlin, Heidelberg: Springer-Verlag; 2020. p. 107–122. https://doi.org/10.1007/978-3-030-58621-8_7, doi: 10.1007/978-3-030-58621-8_7.

Xing E, Jordan M, Russell SJ, Ng A. Distance Metric Learning with Application to Clustering with Side-Information. In: Becker S, Thrun S, Obermayer K, editors. *Advances in Neural Information Processing Systems*, vol. 15 MIT Press; 2002. https://proceedings.neurips.cc/paper_files/paper/2002/file/c3e4035af2a1cde9f21e1ae1951ac80b-Paper.pdf.

Yang F, Chang X, Dang C, Zheng Z, Sakti S, Nakamura S, Wu Y. ReMOTS: Self-Supervised Refining Multi-Object Tracking and Segmentation. ArXiv. 2020; abs/2007.03200. https://api.semanticscholar.org/CorpusID:220404263.

Yi D, Lei Z, Liao S, Li SZ. Deep Metric Learning for Person Re-identification. In: *2014 22nd International Conference on Pattern Recognition*; 2014. p. 34–39. doi: 10.1109/ICPR.2014.16.

Zhang X, Zhou X, Lin M, Sun J, ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices; 2017.

Supplementary tables

Name	Accuracy without crossings (%)			Accuracy with crossings (%)			Tracking time		
	Median	Mean	20-80 percentiles	Median	Mean	20-80 percentiles	Median	Mean	20-80 percentiles
drosophila_6	99.735	99.272	98.797 - 99.735	99.649	99.218	98.776 - 99.649	0:21:39	0:22:26	0:21:39 - 0:23:15
drosophila_8	99.100	98.829	98.545 - 99.100	98.779	98.503	98.214 - 98.779	0:44:47	0:48:31	0:21:58 - 1:07:37
drosophila_10	99.006	99.006	one point only	99.007	99.007	one point only	6:28:35	6:28:31	6:18:53 - 6:38:12
drosophila_38	99.842	99.875	99.842 - 99.909	99.756	99.798	99.756 - 99.842	6:55:15	6:13:43	4:16:03 - 7:23:33
drosophila_59	99.737	99.737	99.737 - 99.737	99.707	99.707	99.707 - 99.707	1 day, 3h	1 day, 3h	9:41:00 - 1 day, 13h
drosophila_60	99.957	99.957	one point only	99.924	99.924	one point only	4 days, 18h	4 days, 18h	4 days, 15h - 4 days, 21h
drosophila_72	99.974	99.982	99.974 - 99.990	99.967	99.973	99.967 - 99.979	11:29:47	12:57:41	7:00:26 - 16:42:43
drosophila_80	99.628	99.628	one point only	99.558	99.558	one point only	6 days, 15h	6 days, 15h	6 days, 15h - 6 days, 16h
drosophila_100_1	98.268	98.268	one point only	98.108	98.108	one point only	8 days, 1h	8 days, 1h	8 days, 1h - 8 days, 2h
drosophila_100_2	97.606	97.606	one point only	97.583	97.583	one point only	9 days, 22h	9 days, 22h	9 days, 21h - 9 days, 22h
drosophila_100_3	69.658	69.658	one point only	69.432	69.432	one point only	13 days, 14h	13 days, 14h	13 days, 13h - 13 days, 15h
mice_2_1	99.170	99.227	99.170 - 99.285	97.364	97.726	97.364 - 98.088	0:05:47	0:07:12	0:05:47 - 0:08:36
mice_2_2	99.095	99.081	99.067 - 99.095	98.051	98.085	98.051 - 98.119	0:08:08	0:07:19	0:06:30 - 0:08:08
mice_2_3	99.959	97.571	95.140 - 99.959	99.221	97.171	95.084 - 99.221	0:04:58	0:06:00	0:04:58 - 0:07:03
mice_2_4	99.962	99.974	99.962 - 99.985	99.958	99.970	99.958 - 99.982	0:06:19	0:05:23	0:04:28 - 0:06:19
mice_4_1	99.467	99.284	99.098 - 99.467	99.445	99.169	98.891 - 99.445	0:14:27	0:12:37	0:10:45 - 0:14:27
mice_4_2	93.771	92.112	90.411 - 93.771	93.768	92.098	90.386 - 93.768	0:13:58	0:14:56	0:13:58 - 0:15:55
zebrafish_5	99.687	99.822	99.687 - 99.961	99.657	99.804	99.657 - 99.955	0:08:40	0:11:35	0:08:40 - 0:14:36
zebrafish_7	99.994	99.982	99.970 - 99.994	99.973	99.963	99.953 - 99.973	0:13:59	0:15:39	0:13:59 - 0:17:22
zebrafish_10_1	99.999	99.999	99.999 - 99.999	99.997	99.997	99.997 - 99.999	0:44:31	0:44:12	0:42:42 - 0:45:24
zebrafish_10_2	99.987	99.987	99.987 - 99.987	99.982	99.978	99.975 - 99.982	0:47:18	0:47:23	0:47:18 - 0:47:27
zebrafish_10_3	99.992	99.992	99.992 - 99.992	99.991	99.987	99.984 - 99.991	0:45:08	0:44:05	0:43:02 - 0:45:08
zebrafish_10_4	99.965	99.978	99.965 - 99.990	99.964	99.977	99.964 - 99.989	0:20:05	0:19:19	0:18:32 - 0:20:05
zebrafish_20	99.997	99.998	99.997 - 99.999	99.981	99.980	99.979 - 99.981	1:01:06	0:56:18	0:51:24 - 1:01:06
zebrafish_60_1	99.332	99.306	99.279 - 99.332	99.333	99.307	99.281 - 99.333	3:03:42	2:54:11	2:44:27 - 3:03:42
zebrafish_60_2	99.901	99.939	99.901 - 99.977	99.887	99.927	99.887 - 99.967	3:58:33	3:01:47	2:04:41 - 3:58:33
zebrafish_60_3	99.990	99.990	99.989 - 99.990	99.988	99.987	99.987 - 99.988	2:27:01	2:34:42	2:27:01 - 2:42:29
zebrafish_80_1	99.849	99.905	99.815 - 99.999	99.818	99.888	99.790 - 99.991	8:30:00	12:13:23	6:29:28 - 11:29:21
zebrafish_80_2	99.950	99.974	99.950 - 99.997	99.948	99.971	99.948 - 99.994	4:17:00	4:19:08	4:17:00 - 4:21:16
zebrafish_80_3	99.999	99.973	99.947 - 99.999	99.992	99.967	99.941 - 99.992	3:37:08	4:21:01	3:37:08 - 5:05:52
zebrafish_100_1	94.468	97.195	94.468 - 99.927	94.051	96.976	94.051 - 99.907	11:55:32	12:01:12	10:12:42 - 13:06:53
zebrafish_100_2	99.984	99.982	99.981 - 99.984	99.978	99.977	99.976 - 99.978	5:25:35	5:34:52	5:25:35 - 5:44:14
zebrafish_100_3	99.966	99.952	99.937 - 99.966	99.961	99.947	99.933 - 99.961	6:20:55	6:01:33	5:41:37 - 6:20:55

Supplementary Table 1. Performance of original idtracker.ai (v4) in the benchmark.

Name	Accuracy without crossings (%)			Accuracy with crossings (%)			Tracking time		
	Median	Mean	20-80 percentiles	Median	Mean	20-80 percentiles	Median	Mean	20-80 percentiles
drosophila_6	98.449	98.467	97.720 - 99.194	98.209	98.353	97.753 - 99.081	0:04:28	0:04:38	0:04:26 - 0:05:13
drosophila_8	99.100	99.294	99.100 - 100.000	98.779	99.032	98.769 - 99.961	0:04:46	0:04:59	0:03:39 - 0:05:14
drosophila_10	99.950	98.864	96.805 - 99.950	99.950	98.864	96.804 - 99.950	1:36:47	1:15:54	0:31:15 - 1:43:01
drosophila_38	99.997	99.987	99.993 - 99.998	99.959	99.948	99.951 - 99.977	0:30:22	0:31:29	0:23:22 - 0:37:52
drosophila_59	99.997	99.933	99.861 - 100.000	99.984	99.926	99.856 - 99.997	1:42:52	1:59:44	1:13:52 - 2:28:11
drosophila_60	100.000	99.966	99.885 - 100.000	100.000	99.954	99.825 - 100.000	1 day, 14h	1 day, 5h	3:54:48 - 1 day, 15h
drosophila_72	99.990	99.992	99.989 - 99.997	99.982	99.984	99.981 - 99.990	1:06:10	1:20:44	0:37:58 - 1:35:37
drosophila_80	99.942	99.941	99.921 - 99.960	99.838	99.835	99.816 - 99.845	2:12:26	5:01:52	1:24:11 - 3:26:08
drosophila_100_1	99.944	99.911	99.820 - 99.970	99.834	99.799	99.716 - 99.844	2:51:26	10:36:07	1:45:28 - 1 day, 2h
drosophila_100_2	98.937	98.937	one point only	98.916	98.916	one point only	1 day, 18h	1 day, 18h	1 day, 17h - 1 day, 19h
drosophila_100_3	99.833	99.818	99.720 - 99.859	99.763	99.746	99.655 - 99.786	2:43:07	7:54:35	1:39:37 - 4:09:37
mice_2_1	99.659	99.674	99.616 - 99.707	98.368	98.355	97.893 - 98.699	0:02:15	0:02:21	0:02:14 - 0:02:29
mice_2_2	99.176	99.162	99.053 - 99.179	97.881	97.968	97.687 - 97.987	0:02:50	0:02:49	0:02:34 - 0:03:01
mice_2_3	99.883	98.778	94.339 - 99.965	99.043	98.028	93.784 - 99.168	0:03:05	0:03:08	0:02:39 - 0:03:39
mice_2_4	99.937	99.934	99.937 - 99.937	99.890	99.872	99.828 - 99.895	0:02:04	0:02:07	0:01:48 - 0:02:16
mice_4_1	99.883	99.716	99.300 - 99.973	99.779	99.511	98.720 - 99.800	0:04:36	0:04:58	0:04:34 - 0:06:53
mice_4_2	93.239	93.401	93.226 - 93.891	93.189	93.327	93.185 - 93.808	0:06:46	0:08:31	0:04:37 - 0:12:06
zebrafish_5	99.999	99.998	99.994 - 99.999	99.991	99.991	99.988 - 99.991	0:01:51	0:01:49	0:01:40 - 0:01:53
zebrafish_7	99.968	99.762	99.391 - 99.992	99.940	99.745	99.379 - 99.971	0:02:29	0:03:11	0:02:22 - 0:05:05
zebrafish_10_1	100.000	100.000	100.000 - 100.000	100.000	100.000	100.000 - 100.000	0:08:50	0:08:47	0:08:32 - 0:09:09
zebrafish_10_2	100.000	99.953	99.769 - 100.000	99.995	99.947	99.757 - 99.996	0:09:24	0:09:51	0:08:48 - 0:11:48
zebrafish_10_3	100.000	99.727	100.000 - 100.000	99.997	99.729	99.996 - 99.998	0:08:54	0:09:15	0:08:32 - 0:08:54
zebrafish_10_4	100.000	99.998	99.998 - 100.000	99.999	99.997	99.996 - 99.999	0:02:52	0:02:55	0:02:50 - 0:03:10
zebrafish_20	99.998	99.996	99.988 - 99.999	99.950	99.953	99.948 - 99.966	0:08:29	0:08:37	0:07:26 - 0:10:46
zebrafish_60_1	99.999	99.999	99.999 - 100.000	99.997	99.997	99.995 - 99.998	0:21:14	0:21:34	0:20:34 - 0:24:17
zebrafish_60_2	99.989	99.947	99.973 - 99.999	99.978	99.939	99.962 - 99.996	0:37:40	0:33:21	0:20:29 - 0:43:23
zebrafish_60_3	99.983	99.982	99.968 - 99.999	99.964	99.967	99.951 - 99.995	0:31:47	0:35:34	0:30:53 - 0:40:43
zebrafish_80_1	99.999	99.993	99.999 - 100.000	99.991	99.985	99.991 - 99.992	1:00:22	1:25:46	0:47:00 - 1:22:09
zebrafish_80_2	99.989	99.985	99.961 - 99.999	99.985	99.981	99.957 - 99.995	0:30:23	0:29:18	0:27:24 - 0:30:27
zebrafish_80_3	100.000	100.000	100.000 - 100.000	99.992	99.992	99.992 - 99.995	0:30:52	0:55:01	0:27:00 - 1:29:25
zebrafish_100_1	99.998	99.994	99.983 - 99.998	99.977	99.976	99.969 - 99.978	1:46:04	1:48:59	1:30:47 - 2:09:56
zebrafish_100_2	99.998	99.988	99.954 - 99.998	99.992	99.983	99.951 - 99.995	0:45:27	0:46:28	0:43:36 - 0:56:34
zebrafish_100_3	99.999	99.988	99.942 - 100.000	99.994	99.983	99.936 - 99.995	0:59:47	1:04:01	0:56:01 - 1:43:51

Supplementary Table 2. Performance of optimized v4 (v5) in the benchmark.

	Accuracy without crossings (%)			Accuracy with crossings (%)			Tracking time		
Name	**Median**	**Mean**	**20-80 percentiles**	**Median**	**Mean**	**20-80 percentiles**	**Median**	**Mean**	**20-80 percentiles**
drosophila_6	99.991	99.955	99.802 - 99.997	99.975	99.933	99.764 - 99.976	0:02:12	0:02:15	0:02:10 - 0:02:35
drosophila_8	100.000	99.999	99.997 - 100.000	99.961	99.932	99.818 - 99.961	0:01:53	0:01:50	0:01:39 - 0:01:54
drosophila_10	99.622	99.436	99.211 - 99.911	99.622	99.435	99.212 - 99.911	0:07:28	0:07:27	0:07:21 - 0:07:33
drosophila_38	99.996	99.986	99.950 - 99.996	99.928	99.928	99.903 - 99.940	0:12:04	0:12:01	0:11:55 - 0:12:09
drosophila_59	99.929	99.900	99.916 - 99.945	99.921	99.890	99.894 - 99.941	0:18:08	0:28:40	0:16:08 - 0:33:46
drosophila_60	99.967	99.766	99.024 - 99.968	99.916	99.727	98.992 - 99.934	0:13:58	0:14:55	0:13:30 - 0:19:16
drosophila_72	99.998	99.998	99.997 - 100.000	99.992	99.992	99.989 - 99.994	0:19:44	0:19:18	0:18:45 - 0:20:03
drosophila_80	99.922	99.655	99.889 - 99.966	99.852	99.599	99.848 - 99.929	0:23:24	0:26:20	0:23:05 - 0:23:57
drosophila_100_1	99.936	99.944	99.936 - 99.948	99.864	99.879	99.860 - 99.886	0:22:58	0:22:49	0:21:07 - 0:23:37
drosophila_100_2	99.999	99.764	98.837 - 100.000	99.985	99.749	98.818 - 99.988	0:32:32	0:38:42	0:31:19 - 1:03:00
drosophila_100_3	99.379	99.441	99.079 - 99.574	99.273	99.362	99.006 - 99.510	0:27:30	0:27:24	0:26:38 - 0:27:59
mice_2_1	99.890	99.888	99.881 - 99.902	99.219	99.281	99.177 - 99.469	0:03:15	0:03:13	0:03:10 - 0:03:29
mice_2_2	99.886	99.860	99.818 - 99.922	98.680	98.546	98.408 - 98.995	0:03:11	0:03:14	0:03:09 - 0:03:35
mice_2_3	100.000	100.000	100.000 - 100.000	99.341	99.360	99.303 - 99.478	0:03:03	0:03:01	0:02:41 - 0:03:04
mice_2_4	100.000	100.000	100.000 - 100.000	99.966	99.966	99.962 - 99.971	0:02:53	0:02:51	0:02:39 - 0:03:06
mice_4_1	99.946	99.924	99.935 - 99.955	99.801	99.796	99.799 - 99.856	0:03:19	0:03:11	0:02:43 - 0:03:25
mice_4_2	99.150	99.073	98.418 - 99.907	98.913	98.970	98.445 - 99.794	0:03:33	0:03:46	0:03:19 - 0:04:58
zebrafish_5	99.999	99.999	99.998 - 99.999	99.992	99.990	99.986 - 99.995	0:01:23	0:01:23	0:01:21 - 0:01:25
zebrafish_7	99.970	99.969	99.969 - 99.971	99.945	99.944	99.944 - 99.948	0:02:02	0:02:05	0:01:56 - 0:02:13
zebrafish_10_1	100.000	100.000	99.999 - 100.000	100.000	100.000	99.999 - 100.000	0:08:37	0:08:51	0:08:32 - 0:09:22
zebrafish_10_2	100.000	100.000	100.000 - 100.000	99.996	99.995	99.996 - 99.997	0:09:47	0:09:48	0:09:40 - 0:10:03
zebrafish_10_3	100.000	100.000	99.999 - 100.000	99.998	99.998	99.997 - 99.999	0:11:32	0:11:34	0:11:11 - 0:11:38
zebrafish_10_4	99.999	99.997	99.996 - 100.000	99.995	99.995	99.994 - 99.999	0:02:08	0:02:06	0:01:58 - 0:02:09
zebrafish_20	99.999	99.997	99.999 - 99.999	99.956	99.957	99.955 - 99.966	0:03:47	0:03:42	0:03:38 - 0:03:50
zebrafish_60_1	99.982	99.989	99.982 - 99.999	99.980	99.987	99.980 - 99.997	0:20:40	0:21:17	0:20:36 - 0:22:24
zebrafish_60_2	99.989	99.984	99.949 - 99.997	99.987	99.977	99.938 - 99.989	0:24:20	0:23:31	0:23:43 - 0:24:34
zebrafish_60_3	99.999	99.988	99.946 - 99.999	99.987	99.978	99.934 - 99.988	0:26:58	0:26:57	0:26:46 - 0:27:13
zebrafish_80_1	99.999	99.999	99.998 - 99.999	99.994	99.994	99.994 - 99.994	0:29:02	0:29:48	0:28:22 - 0:30:15
zebrafish_80_2	99.989	99.990	99.978 - 99.998	99.987	99.989	99.976 - 99.997	0:29:50	0:29:31	0:28:08 - 0:31:02
zebrafish_80_3	99.997	99.980	99.989 - 99.999	99.991	99.975	99.982 - 99.992	0:29:15	0:29:32	0:28:25 - 0:30:45
zebrafish_100_1	99.993	99.991	99.983 - 99.999	99.983	99.980	99.973 - 99.986	0:42:54	0:42:59	0:42:46 - 0:44:21
zebrafish_100_2	99.955	99.965	99.955 - 99.998	99.952	99.962	99.952 - 99.995	0:37:33	0:37:31	0:35:35 - 0:37:58
zebrafish_100_3	99.988	99.979	99.987 - 99.995	99.984	99.975	99.983 - 99.990	0:36:18	0:36:34	0:36:05 - 0:37:47

Supplementary Table 3. Performance of new idtracker.ai (v6) in the benchmark.

Name	Accuracy without crossings (%)			Accuracy with crossings (%)			Tracking time		
	Median	Mean	20-80 percentiles	Median	Mean	20-80 percentiles	Median	Mean	20-80 percentiles
drosophila_6	99.943	99.921	99.859 - 99.974	99.878	99.849	99.729 - 99.964	0:26:55	0:27:12	0:21:45 - 0:32:32
drosophila_8	94.665	94.231	87.833 - 99.994	94.680	94.110	87.583 - 99.988	0:31:08	0:31:26	0:28:55 - 0:35:34
drosophila_10	97.934	98.339	97.899 - 97.940	97.936	98.340	97.901 - 97.942	0:37:17	0:37:07	0:36:32 - 0:40:18
drosophila_38	99.915	99.838	99.667 - 99.916	99.807	99.678	99.372 - 99.842	1:34:29	1:21:28	1:25:16 - 1:42:35
drosophila_59	99.862	99.364	97.554 - 99.874	99.824	99.348	97.556 - 99.873	0:52:17	0:59:37	0:22:26 - 1:27:04
drosophila_60	99.998	99.987	99.943 - 99.999	99.998	99.987	99.943 - 99.999	1:05:04	1:10:06	1:00:46 - 1:31:12
drosophila_72	97.793	96.381	91.004 - 99.971	97.789	96.374	90.994 - 99.968	1:02:38	1:16:05	0:49:20 - 1:49:13
drosophila_80	97.643	97.750	95.803 - 99.230	97.618	97.709	95.736 - 99.184	1:39:25	1:38:16	1:20:36 - 1:49:03
drosophila_100_1	99.955	97.300	92.467 - 99.959	99.954	97.280	92.430 - 99.958	2:08:46	1:47:50	0:55:32 - 2:21:53
drosophila_100_2	74.958	73.656	60.167 - 85.481	74.932	73.648	60.163 - 85.470	0:46:31	0:50:55	0:44:09 - 0:53:05
drosophila_100_3	94.200	93.859	91.874 - 96.935	94.123	93.772	91.744 - 96.836	1:02:03	1:12:59	1:01:52 - 1:19:57
mice_2_1	99.683	99.419	99.611 - 99.761	98.515	98.141	97.791 - 98.597	0:05:46	0:05:40	0:05:36 - 0:05:52
mice_2_2	99.118	98.257	95.102 - 99.550	96.376	94.821	91.215 - 96.897	0:04:06	0:04:04	0:03:22 - 0:04:29
mice_2_3	99.775	99.298	98.319 - 99.801	97.601	96.928	95.375 - 98.185	0:04:03	0:04:04	0:04:01 - 0:04:15
mice_2_4	95.925	95.708	95.793 - 96.043	95.075	94.812	94.757 - 95.463	0:02:45	0:03:25	0:02:03 - 0:05:11
mice_4_1	99.964	99.959	99.940 - 99.965	99.577	99.574	99.562 - 99.581	0:18:11	0:18:10	0:18:10 - 0:18:31
mice_4_2	93.100	92.243	92.480 - 93.523	92.680	91.795	91.818 - 93.283	0:20:34	0:18:38	0:16:13 - 0:25:05
zebrafish_5	100.000	100.000	99.999 - 100.000	100.000	100.000	99.999 - 100.000	0:09:57	0:09:48	0:09:06 - 0:10:39
zebrafish_7	99.982	99.987	99.981 - 99.998	99.981	99.985	99.979 - 99.998	0:08:37	0:09:01	0:06:41 - 0:10:16
zebrafish_10_1	99.864	99.776	99.355 - 99.890	99.852	99.770	99.347 - 99.891	0:13:57	0:13:44	0:13:30 - 0:14:43
zebrafish_10_2	99.972	99.928	99.913 - 99.993	99.968	99.924	99.907 - 99.987	0:22:07	0:21:58	0:21:03 - 0:23:23
zebrafish_10_3	99.869	99.647	99.684 - 99.998	99.861	99.639	99.666 - 99.997	0:37:51	0:31:48	0:19:06 - 0:40:01
zebrafish_10_4	99.998	99.996	99.998 - 99.998	99.996	99.994	99.995 - 99.998	0:14:12	0:14:23	0:12:36 - 0:15:16
zebrafish_20	99.942	99.872	99.717 - 99.988	99.845	99.841	99.717 - 99.959	0:43:02	0:48:17	0:36:57 - 1:03:46
zebrafish_60_1	99.887	99.920	99.873 - 99.985	99.881	99.915	99.868 - 99.981	1:43:58	1:43:05	1:34:08 - 1:46:02
zebrafish_60_2	99.541	98.755	95.295 - 99.674	99.520	98.737	95.268 - 99.657	1:30:15	1:20:09	0:42:10 - 1:35:34
zebrafish_60_3	99.229	99.357	99.091 - 99.795	99.223	99.352	99.090 - 99.790	1:39:17	1:31:10	0:59:17 - 1:45:48
zebrafish_80_1	99.655	99.696	99.628 - 99.657	99.645	99.686	99.618 - 99.646	1:23:48	1:26:36	1:20:59 - 1:35:23
zebrafish_80_2	99.689	99.686	99.595 - 99.722	99.688	99.683	99.587 - 99.720	1:38:38	1:36:16	1:26:50 - 1:40:29
zebrafish_80_3	99.789	99.650	99.719 - 99.977	99.782	99.641	99.713 - 99.972	1:36:15	1:34:10	1:24:55 - 1:36:51
zebrafish_100_1	99.565	99.052	98.559 - 99.731	99.547	99.022	98.540 - 99.703	1:36:42	1:20:06	0:59:43 - 1:57:50
zebrafish_100_2	97.988	98.334	97.751 - 99.837	97.976	98.323	97.734 - 99.832	1:06:55	1:10:49	1:04:29 - 2:02:34
zebrafish_100_3	99.293	98.675	95.432 - 99.836	99.286	98.664	95.401 - 99.833	1:15:39	1:13:24	0:22:01 - 1:42:34

Supplementary Table 4. Performance of TRex in the benchmark.

Appendix 1

Computation of tracking accuracy

We used the idtracker.ai Validator tool (see ***Appendix 4***) to manually generate ground-truth trajectories. The input to the Validator were outputs from idtracker.ai v5, and the Validator facilitates that the user manually corrects the mistakes. The ground-truth obtained includes the position and identity of each animal in every frame, along with their classification as either individual or crossing.

Tracking accuracy is quantified using the standard Identification F1 Score (IDF1), defined in ***Ristani et al.*** (***2016***) as:

$$\text{IDF1} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{1}$$

Here, TP (true positive) refers to predicted positions that match their ground-truth points within a distance T. FP (false positive) are predicted positions with no corresponding ground-truth match within a distance T, and FN (false negative) are ground-truth positions with no matching prediction within a distance T.

An identity switch in a single frame results in two false positives and two false negatives, as both predicted and ground-truth identities become unmatched. False negatives also occur when the software either fails to detect an animal or loses track of its identity in a given frame.

We rely on IDF1 rather than other multi-object tracking metrics such as MOTA or MOTP ***Bernardin and Stiefelhagen*** (***2008***), since the central goal of our system is to maintain consistent identities across time. MOTA (Multiple Object Tracking Accuracy) counts an identity switch as a single error regardless of its persistence, thereby underestimating the severity of prolonged misidentifications. MOTP (Multiple Object Tracking Precision) evaluates how well predicted positions or bounding boxes align with ground-truth locations, focusing on spatial localization accuracy. While useful for detection-centric benchmarks, neither MOTA nor MOTP adequately capture the temporal consistency of identities. In contrast, IDF1 directly reflects whether trajectories preserve correct identity assignments throughout the video, which is the relevant criterion for evaluating our system.

We report two types of accuracy: **accuracy with crossings**, which includes all trajectory points; and **accuracy without crossings**, where points labeled as crossings in the ground-truth are excluded from the evaluation.

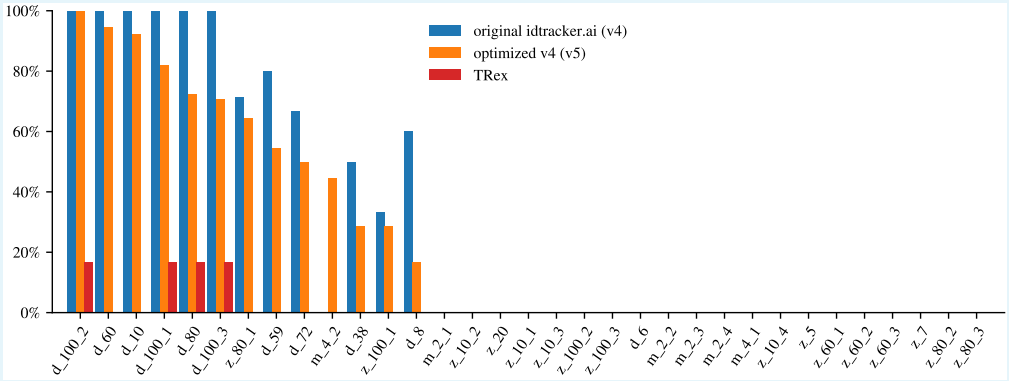
We present all results using $T = 1$BL with BL being a body length. We also verified that accuracy remains largely unaffected by the value of T. For instance, reducing it to $T = 0.5$BL results in a very small change of the mean accuracy (without crossings) across the benchmark in the new idtracker.ai from 99.923% to 99.908%.

Benchmark of accuracy and tracking time

To evaluate the tracking time and accuracy of version 4 (4.0.12), 5 (5.2.12) and 6 (6.0.9) of idtracker.ai and version 1.1.9 of TRex, we used a set of 33 videos with their corresponding human-validated ground-truth trajectories. Each video is 10 minutes long, has a frame rate between 25 and 60 fps and features one of three species: mice, drosophila, or zebrafish, with the number of individuals ranging from 2 to 100. Crossing blobs in these videos make, on average, the 2.6% of the total amount of blobs (1.1% for zebrafish, 0.7% for drosophila, and 9.4% for mice videos).

Both idtracker.ai and TRex rely on several tracking parameters. Some of them have default values that typically require no adjustment (e.g., minimum blob size, network and training hyperparameters) or have a single correct value (e.g., number of animals). In contrast,

variable parameters do not have a unique correct value but rather a valid range, from which the user must select a specific value to run the system. In idtracker.ai, the only variable parameter is `intensity_threshold`, whereas in TRex two such parameters exist: `threshold` and `track_max_speed`.



Appendix 1—figure 1. Protocol 2 failure rate. Probability for the different tracking systems of not tracking the video with Protocol 2 in idtracker.ai (v4 and v5) and in TRex the probability that it fails without generating trajectories.

For the variable parameters, choosing one value or another within the valid interval can give different tracking results. For some values, previous versions of idtracker.ai (v4 and v5) can stochastically resort to a protocol for tracking that we call protocol 3 (***Appendix 1—Figure 1***), a method that can take days to process more complex videos. Similarly, TRex can crash without outputting any trajectory in certain videos, leading to missing accuracy outputs (***Appendix 1—Figure 1***).

For this reason, simulating a user that, for a given video, runs the tracking software a single time can result in extremely slow tracking for previous versions of idtracker.ai and a failed tracking for TRex. This would give an advantage to the new version v6 of idtracker.ai since it does not have the slow protocol 3 and never crashed in our tests. We thus simulated a more realistic user that, out of up to 5 attempts, uses the first run in which there is no use of the slow protocol 3 in idtracker.ai v4 and v5 (or protocol 3 is used if it consumes the 5 runs) and in TRex the first run in which the software does not crash. The tracking times are then the sum of tracking times of the attempts used.

To simulate our user many times, we first prepared the dataset of tracking runs by repeating the tracking for each video and software until reaching 5 successful runs or a maximum of 35 attempts. For the original version of idtracker.ai, this was limited to 3 successful runs or 7 attempts due to significantly longer tracking times. Each new run uses new random values for the variable parameters sampled from a precomputed interval we consider an expert user can chose from. In successful runs, both accuracy and tracking time are recorded. In failed runs, when idtracker.ai switches to protocol 3 or TRex crashes, only the time until failure is recorded.

We then simulated our user up to 10,000 times per software and video by sampling tracking runs from our dataset of tracking runs to obtain robust estimates of the tracking times and accuracies. ***Figure 1*** and ***Figure 1—figure Supplement 1*** report the median accuracies, without and with crossings, respectively, and tracking times. ***Supplementary Table 1***, ***Supplementary Table 2***, ***Supplementary Table 3***, and ***Supplementary Table 4*** present the median, mean, and the 20 and 80 percentiles in v4, v5, v6 and TRex respectively.

To ensure a fair comparison, TGrabs (a video pre-processing tool needed by TRex) is included when running TRex, graphical interfaces are always disabled at runtime to maximize

performance, `output_interpolate_positions` is enabled in TRex and posture estimation was not deactivated nor used in TRex.

Appendix 2

Improvements to the original idtracker.ai in version 5

Following the last publication of idtracker.ai ***Romero-Ferrero et al.*** (***2019***), the software underwent continuous maintenance, including feature additions, performance optimizations, and hyperparameter tuning (released via PyPI from March 2023 for v5.0.0 to June 2024 for v5.2.12). These updates improved the implementation and tracking pipeline but did not alter the core algorithm. Significant advancements were made in user experience, tool availability, processing speed, and memory efficiency. Below, we summarize the most notable changes.

Blob memory optimization: Blobs are defined as collections of connected pixels belonging to one or more animals. In v4, blobs stored pixel indices, causing memory usage to scale quadratically with blob size. In v5, blobs are represented by simplified contours using the Teh-Chin chain approximation ***Teh and Chin*** (***1989***), reducing memory usage by 93% in blob instances. This also accelerated blob-related computations (centroid, orientation, area, overlap, identification image creation, etc.).

Efficient image loading: Identification images are now efficiently loaded on demand from HDF5 files, eliminating the need to load all images into memory. This enables training with all images regardless of video length, with minimal memory usage.

Code optimization: The source code was revised to eliminate speed bottlenecks. The most impactful changes include:

- Frame segmentation accelerated by 80% through optimized OpenCV usage.
- Faster blob-to-blob overlap checks by first evaluating bounding boxes before deeper comparisons.
- Persistent storage of blob overlap checks to avoid redundant computations when reloading data.
- Efficient disk access for identification images by reading them in sorted batches, minimizing I/O overhead.
- Reduced bounding box image sizes to the minimum necessary, lowering memory and processing demands.
- Optimized and parallelized Torch data loaders for more efficient model training.
- Caching of computationally expensive properties for blobs, fragments, and global fragments.
- Sorted fragment lists to speed up coexistence detection.

Changes to the identification protocol: In v4, identity assignments to high-confidence fragments were fixed and excluded from downstream correction, regardless of later evidence. In v5, this was relaxed for short fragments (fewer than 4 frames), allowing corrections due to their statistical unreliability and frequent image noise.

Expanded idtracker.ai ecosystem. A restructure of the main graphic user interface and the creation of a new validation app for inspecting and correcting tracking results. Direct integration of idmatcher.ai, originally introduced in ***Romero-Ferrero et al.*** (***2023***), that allows propagation of consistent identity labels across multiple recordings. See ***Appendix 4*** for a full guide on the current extra features of idtracker.ai.

Appendix 3

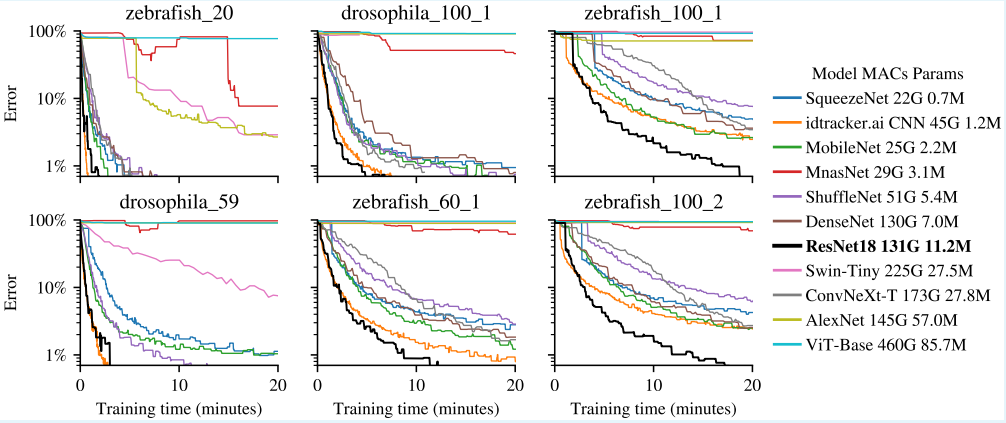
We direct the reader to Appendix B of ***Romero-Ferrero et al.*** (***2019***) for the formal definitions of the main objects and algorithms used in old idtracker.ai, which are common to versions 4, 5, and 6:

- **B.2.1** Segmentation
- **B.2.2** Detection of individual and crossing images
- **B.2.3** Fragmentation
- **B.2.5** Residual identification
- **B.2.6** Post-processing
- **B.2.7** Output

The following sections describe in detail the new identification algorithm introduced in the new idtracker.ai (v6).

Network Architecture

Training represents 47% of the total tracking time in our benchmark. To identify the fastest architecture in training, we evaluated 31 models from 10 families of state-of-the-art CNNs and vision transformers, including the CNN used in versions 4 and 5 of idtracker.ai.

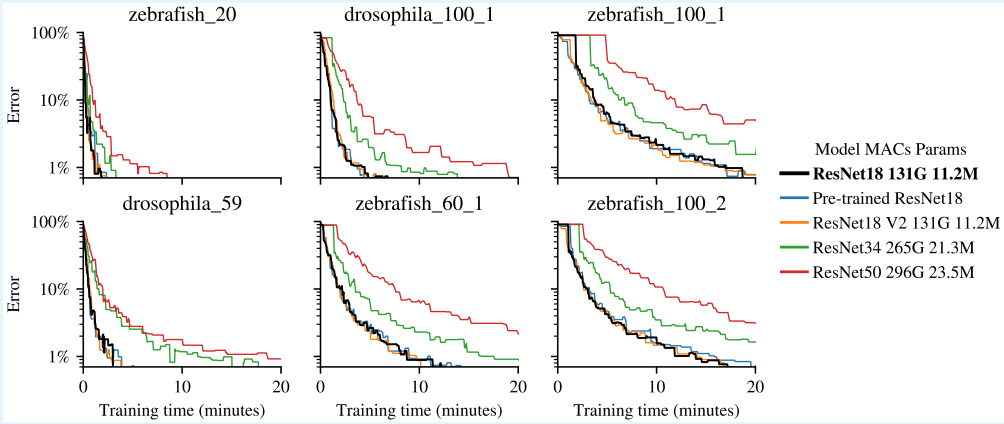


Appendix 3—figure 1. Models comparison. Error in image identification as a function of training time for different deep learning models randomly initialized in 6 test videos. For each network we report the multiply-accumulate operations (MAC) in giga operations (G) (for a batch of 1600 images of size 40x40x1) and the number of parameters in the units of million parameters (M). Every 100 training batches, we perform k-means clustering on a randomly selected set of 20,000 images, assigning identities based on clusters. We then compute the Silhouette score and ground-truth error on the same set. The reported error corresponds to the model with the best Silhouette score observed up to that point.

Within each family, we tested different network sizes and dropout values and plotted the learning curves of the best candidates (***Appendix 3—Figure 1***). The earlier idtracker.ai CNN performed adequately on simpler videos (e.g., *zebrafish_20*) but struggled in more challenging ones (e.g., *zebrafish_100_2*). Vision transformers (ViT ***Dosovitskiy et al.*** (***2020***), Swin ***Liu et al.*** (***2021***)) converged much more slowly, and in some cases could not be trained due to excessive VRAM requirements. Note that image size typically ranges between 20×20 and 100×100 pixels, and each batch consists of 400 positive and 400 negative image pairs (1,600 images per batch).

We also tested more advanced CNN architectures such as SqueezeNet ***Iandola et al.*** (***2016***), MobileNet ***Howard et al.*** (***2017***), MnasNet ***Tan et al.*** (***2018***), ShuffleNet ***Zhang et al.*** (***2017***), DenseNet ***Huang et al.*** (***2016***), and AlexNet ***Krizhevsky*** (***2014***). None achieved the

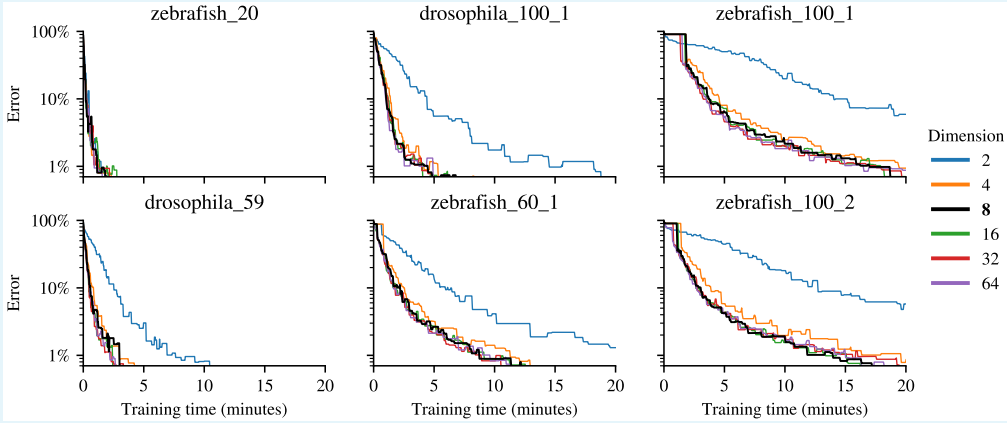
same error levels within comparable training times as our baseline CNN. In contrast, ResNet ***He et al.*** (***2016a***) consistently outperformed all other models, offering the best balance between training speed and accuracy across videos.



Appendix 3—figure 2. ResNet models comparison. Error in image identification as a function of training time for different deep learning models randomly initialized (except *Pre-trained ResNet18*) in 6 test videos. For each network we report the multiply-accumulate operations (MAC) in giga operations (G) (for a batch of 1600 images of size 40x40x1) and the number of parameters in the units of million parameters (M). Every 100 training batches, we perform k-means clustering on a randomly selected set of 20,000 images, assigning identities based on clusters. We then compute the Silhouette score and ground-truth error on the same set. The reported error corresponds to the model with the best Silhouette score observed up to that point.

Within the ResNet family (***Appendix 3—Figure 2***), the smallest ResNet18 proved optimal: the largest variants (ResNet101, ResNet152) demanded excessive VRAM and the intermediate sizes (ResNet34 and ResNet50) showed slower convergence. Both v1 and v2 versions ***He et al.*** (***2016b***) performed equivalently. Pre-training on ImageNet conferred no benefit, likely due to the domain mismatch between natural images and video-specific animal crops.

Embedding dimension was another key hyperparameter. Here, too, there is a trade-off between achieving a robust representation of subtle differences between animals and maintaining a compact network size and efficient training speed. Empirically, an embedding dimension of 8 provided a good trade-off with other values performing similarly (***Appendix 3—Figure 3***).



Appendix 3—figure 3. Embedding dimensions comparison. Error in image identification as a function of training time for different embedding dimensions in 6 test videos. Every 100 training batches, we perform k-means clustering on a randomly selected set of 20,000 images, assigning identities based on clusters. We then compute the Silhouette score and ground-truth error on the same set. The reported error corresponds to the model with the best Silhouette score observed up to that point.

The final *contrastive learning* network (***Figure 2b***) is a ResNet18 (v1) with a single-channel input for grayscale images and an 8-unit fully connected output layer with no bias and using the identity as activation function. Networks are randomly initialized unless the user indicates to copy the weights from a previous tracking session (see ***Appendix 4*** for a usage example). Training is performed with the Adam optimizer ***Kingma and Ba*** (***2017***) at a learning rate of 0.001.

Loss function

The contrastive loss function operates on pairs of data points, aiming to minimize the distance between positive pairs and maximize the distance for negative pairs. Being F_i a fragment with arbitrary identifier i and I_{ik} and image in the fragment F_i with arbitrary identifier k, the contrastive loss $\mathcal{L}$ for a pair of images (I_{ik}, I_{jl}) is defined as:

$$\begin{aligned}\mathcal{L}\left(I_{ik}, I_{jl}, l_{ik,jl}\right) =& l_{ik,jl} \cdot \max(0, D_{ik,jl} - D_{\text{pos}})^2 \\ &+ (1 - l_{ik,jl}) \cdot \max(0, D_{\text{neg}} - D_{ik,jl})^2\end{aligned} \tag{2}$$

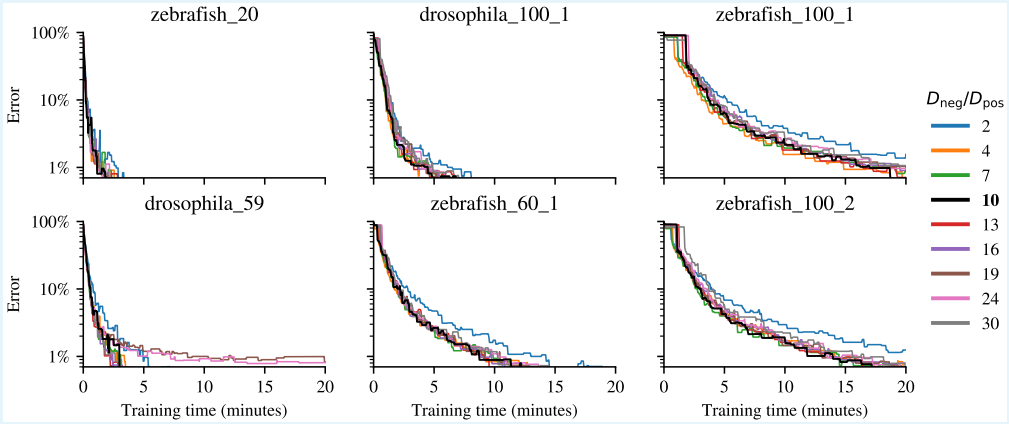
$$l_{ik,jl} = \begin{cases} 1 & \text{if } i = j \text{ (positive pair)} \\ 0 & \text{if } i \neq j \text{ and } F_i \text{ coexists with } F_j \text{ (negative pair)} \end{cases}$$

where $D_{ik,jl}$ is the Euclidean distance between the embedding of I_{ik} and I_{jl}, D_{neg} is the minimum allowed distance in a negative pair of images (images coming from coexisting fragments), and D_{pos} is the maximum allowed distance in a positive pair of images (images from the same fragment). It is important to emphasize that the network processes one image at a time, obtaining a single independent point in the representational space for each image. The Euclidean distance between the embeddings for the corresponding pairs of images is computed only afterwards.

D_{neg} and D_{pos} serve as thresholds to regulate distances in the embedding space. D_{neg} prevents images from negative pairs from being pushed indefinitely far apart, while D_{pos} prevents the collapse of images from positive pairs into a single point.

These thresholds D_{pos} and D_{neg} are crucial in our problem, where we aim to embed images of the same identity in similar regions of the representational space. This is because of the following reason. We cannot compare all possible pairs of images and are instead limited to the fragment structure of the video to obtain the labels $l_{ik,jl}$. This limitation means that the loss function does not directly pull together embeddings of the same identity, but rather images from the same fragment. Similarly, the loss does not push apart embeddings of different identities but images from coexisting fragments. D_{pos} helps prevent the collapse of all images from the same fragment to a single point, allowing for the creation of a diffuse region in the representational space where fragments from the same identity are clustered together. D_{neg} prevents excessive scattering, ensuring better compression of the representational space and maintaining the integrity of clusters of images from the same identity.

We use $D_{\text{pos}} = 1$ and $D_{\text{neg}} = 10$. These values were determined empirically to provide effective embeddings and were robust for tracking multiple videos across various species and different numbers of animals (***Appendix 3—Figure 4***).



Appendix 3—figure 4. D_{neg} over D_{pos} comparison. Error in image identification as a function of training time for different ratios of $D_{\text{neg}}/D_{\text{pos}}$ in 6 test videos. Every 100 training batches, we perform k-means clustering on a randomly selected set of 20,000 images, assigning identities based on clusters. We then compute the Silhouette score and ground-truth error on the same set. The reported error corresponds to the model with the best Silhouette score observed up to that point.

Sampling strategy

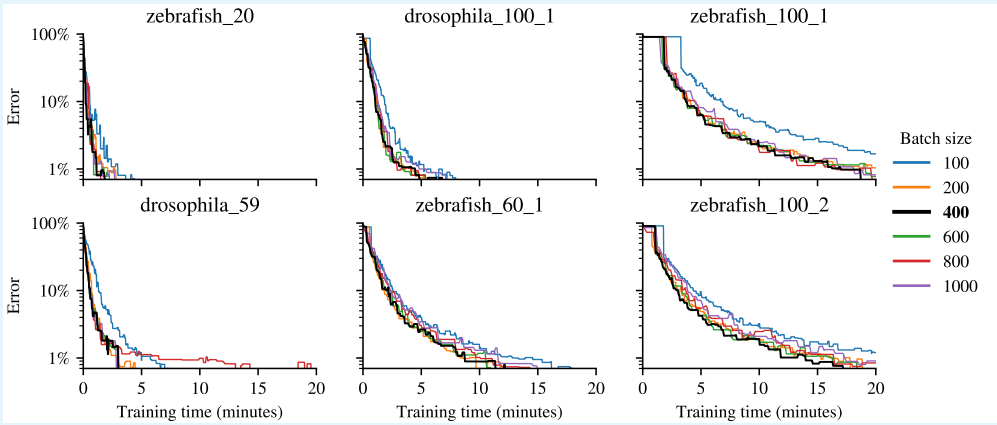
Ideally, we would create two datasets of image pairs: one containing negative pairs and another containing positive pairs. However, the challenge with this approach is that very long videos or those containing a large number of animals can yield trillions of pairs of images, making the process computationally prohibitive. Therefore, we approach the problem with a hierarchical sampling method. For negative pairs, we first select a pair of coexisting fragments, and then we randomly sample an image from each fragment. For a positive pair, we sample two images from the same fragment.

Following this idea, we start by creating two datasets. The first consists of a list of all the fragments in the video, from which we will sample the positive pairs. The second dataset contains all possible pairs of coexisting fragments in the video. From these lists we exclude all fragments smaller than 4 images to reduce possible noisy blobs.

Since the same set of images is used both for training and prediction, and these are already pre-aligned egocentrically, we do not apply data augmentation. Augmenting the data would not introduce new useful information and could, in fact, slow-down the training process. Therefore, the model is trained directly on the unaltered blob images.

Our tests revealed that large and balanced batches, with an equal number of positive and negative pairs, are ideal for our setting of contrastive learning. Specifically, we choose batches consisting of 400 positive pairs of images and 400 negative pairs of images (1600 images in total), as it was the smaller batch size that did not compromise training speed/accuracy (***Appendix 3—Figure 5***). Intuitively, large batch sizes allow for a good spread of pairs from a significant proportion of the video, thereby forcing the network to learn a global embedding of the video. Since positive pairs tend to diminish the size of the representational space while negative pairs tend to increase it, a good balance between the two forces the network to compress the representational space while respecting the negative relationships ***Chen et al.*** (***2020a***). This balance between positive and negative pairs is somewhat surprising, given that several works emphasize the importance of negative examples over positive ones ***Awasthi et al.*** (***2022***); ***Khosla et al.*** (***2021***). While we do not yet have an explanation

for why this balance appears to perform better in our case, we note that it is not possible to compare all images from one class against those of another, as negative pairs of images can only be sampled from coexisting fragments. Additionally, positive pairs that compress the space can only be sampled from the same fragment and not the same identity. Since we cannot compare images freely and are constrained by the fragment structure of the video, we might need more positive pairs to ensure a higher degree of compression of the representational space, such that not only images from the same fragment are close together, but also images from the same identity.



Appendix 3—figure 5. Batch size comparison. Error in image identification as a function of training time for different batch sizes of pairs of images in 6 test videos. Every 100 training batches, we perform k-means clustering on a randomly selected set of 20,000 images, assigning identities based on clusters. We then compute the Silhouette score and ground-truth error on the same set. The reported error corresponds to the model with the best Silhouette score observed up to that point.

The hierarchical sampling allows us to address the question of how to select pairs of fragments to optimize the training speed of the network. Since we sample pairs of fragments rather than directly sampling pairs of images, we need to skew the probability of a pair of fragments being sampled to reflect the number of images they contain. More concretely, let f_i be the number of images in fragment F_i. For negative relations we define $f_{i,j} = f_i + f_j$ and set the probability of sampling the pair F_i, F_j, by their size as:

$$P_s(F_i, F_j) = \frac{f_{i,j}}{\sum_{k=1}^{N-1} \sum_{l=k+1}^{N} f_{k,l}}. \tag{3}$$

For positive pairs, the probability of sampling a given fragment f_i is:

$$P_s(F_i) = \frac{f_i}{\sum_{j=1}^{N} f_j}. \tag{4}$$

By examining the evolution of the clusters during training (***Figure 2c***) it is clear that the learning process is not uniform, as some of the identities become separated sooner than others. ***Figure 2c*** top row second and third columns give us a nice illustration of this phenomenon. The images embedded in the pink rectangle of the representational space already satisfy the loss function, meaning that the negative pairwise relationships are already embedded further away than D_{neg}, and images that form positive pairwise relationships are already embedded closer than D_{pos}. Consequently, the loss function for these pairs is effectively zero, and passing them through the network will not alter the weights, merely prolonging the training process. In contrast, the separation of clusters in the orange square is incomplete, indicating that image pairs in this region still contribute to the loss function. These pairs are more pertinent, as they contain information that the network has yet to

learn. To bias the sampling of image pairs towards those that still contribute to the loss function, each pair of fragments is assigned a loss score. When a pair of images is sampled for training, if the loss for that pair is not zero, the loss score for the corresponding pair of fragments is incremented by one. This score then undergoes an exponential decay of 2% per batch. More specifically, let $l_s(i,j)$ be the loss score of the pair of fragments F_i and F_j, and $\mathcal{L}\left(I_{il}, I_{ik}\right)$ the loss of the images I_{il} and I_{ik}. If the pair I_{il} and I_{ik} is sampled the loss score is updated by

$$l_s(i,j) \longleftarrow \begin{cases} (l_s(i,j)+1)(1-0.02), & \text{if } \mathcal{L}\left(I_{il}, I_{ik}\right) > 0 \\ l_s(i,j)(1-0.02), & \text{otherwise} \end{cases} \tag{5}$$

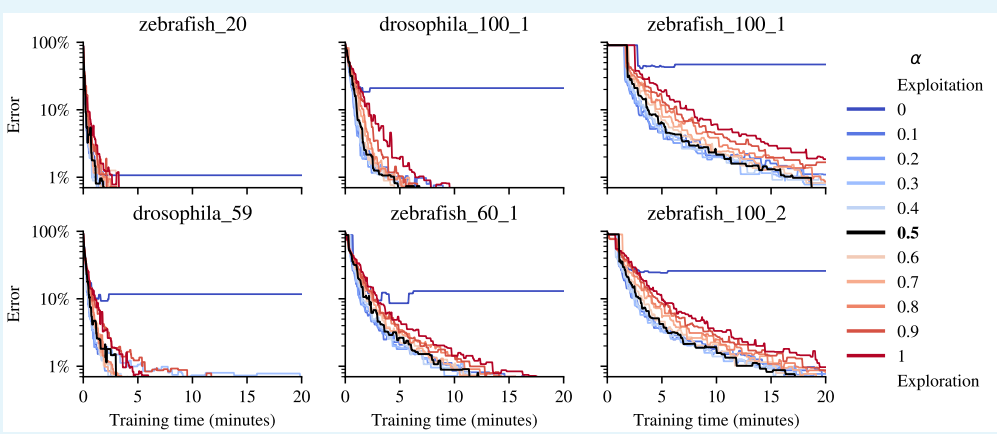
The exponential decay is always applied independently to every pair of fragments, regardless of whether the pairs of images were sampled from those fragments in the previous batch of images or not. The loss score is converted into a probability distribution over all pairs of fragments by

$$P_{l_s}(F_i, F_j) = \begin{cases} \frac{l_s(i,j)}{\sum_{i \neq j} l_s(i,j)}, & \text{if } i \neq j \\ \frac{l_s(i,i)}{\sum_i l_s(i,i)}, & \text{otherwise} \end{cases} \tag{6}$$

The final probability of sampling pairs of fragments is given by

$$P(F_i, F_j) = \alpha P_s(F_i, F_j) + \left(1-\alpha\right) P_{l_s}(F_i, F_j) \tag{7}$$

This balance between these two probabilities can be seen as an exploitation versus exploration paradigm. $P_s(F_i, F_j)$ enforces constant exploration, while $P_{l_s}(F_i, F_j)$ exploits the current state of learning by dynamically updating the sampling probability. This ensures that pairs of fragments containing unlearned knowledge are sampled more frequently, while maintaining a baseline of exploration based on fragment size. We tried several values for α and found that a value of α around $\frac{1}{2}$ produced the best decrease of the time required to train the network across a large collection of videos (***Appendix 3—Figure 6***). It is noteworthy that the failure of the $\alpha = 0$ case renders the contrastive protocol ineffective in solving the tracking problem. This failure occurs because the sampling becomes highly biased towards specific regions of the representational space, leading to only local solutions for the separation of negative pairs and the compression of positive pairs. In effect, the network experiences catastrophic forgetting by focusing excessively on small groups of fragments at a time, thereby compromising the embeddings of other images.



Appendix 3—figure 6. Exploration and exploitation comparison. Error in image identification as a function of training time for different exploration/exploitation weights α in 6 test videos. Every 100 training batches, we perform k-means clustering on a randomly selected set of 20,000 images, assigning identities based on clusters. We then compute the Silhouette score and ground-truth error on the same set. The reported error corresponds to the model with the best Silhouette score observed up to that point.

Clustering and assignment

After training the network using contrastive loss, we pass all images through the network to generate their corresponding embeddings in the learned representational space. These embeddings are then grouped using K-means clustering ***Sculley*** (***2010***). Each cluster ideally represents images of the same identity, as the training process has encouraged the network to place similar images close together and dissimilar ones farther apart in the embedding space. Next, we perform single-image classification, assigning each image a label based on the cluster to which its embedding belongs. Afterwards, the assignment method follows two conditions. If more than half of the images in the video can not be identified and there exist global fragments, the accumulation protocol from the original idtracker.ai is run, see ***Fallback accumulation protocol***. Otherwise, we move straight to residual identification, see ***Residual Identification***.

In order to identify fragments, we not only need an identity prediction for each image but also a probability distribution over all the identities. Let $d_j(I_{ik})$ be the distance of image I_{ik} to the center of cluster j. We define the probability of image I_{ik} belonging to identity j by

$$P\left(I_{ik} \text{ belongs to identity } j\right) = \frac{d_j(I_{ik})^7}{\sum_j d_j(I_{ik})^7} \tag{8}$$

Equation (8) is used to emphasize differences in distances between points and clusters, creating a more peaked probability distribution that clearly distinguishes closer clusters from farther ones. The exponent of 7 smooths the probability distribution and reduces the influence of distant clusters, making the assignment more discriminative. In higher-dimensional spaces like the 8-dimensional space in the paper, distances are more spread out, and using a high power helps to counteract this dispersion, resulting in more confident cluster assignments.

If we are in a scenario where no global fragments exist, K-means is initialized with Greedy k-means++ ***Grunau et al.*** (***2022***). Otherwise, we use the average embedding of the fragment images in one of the global fragments as initial cluster centers. This approach speeds up and stabilizes convergence, allowing us to better compare clusters as training progresses.

Differences with previous work in contrastive/metric learning

Recent advances in representation learning have established powerful, scalable tools for learning visual features without labels. This progress has been spearheaded by contrastive methods such as SimCLR/v2 ***Chen et al.*** (***2020a***,b), SimSiam ***Chen and He*** (***2021***) and Momentum Contrast (MoCo) ***He et al.*** (***2020***), and reconstruction/self-distillation techniques like Masked Autoencoders (MAE) ***He et al.*** (***2022***) and DINOv2 ***Oquab et al.*** (***2024***). Collectively, these frameworks have delivered state-of-the-art results in recognition and feature learning. Their success is demonstrated in specialized domains like human and primate facial recognition ***Schroff et al.*** (***2015***); ***Guo et al.*** (***2020***) as well as in setting new standards for general visual representation quality ***Chen et al.*** (***2020a***,b); ***He et al.*** (***2020***).

The success of these models has also underpinned foundational progress in Multiple Object Tracking (MOT) via Re-Identification (ReID). Their influence is seen in the evolution of ReID-driven tracking, from early Deep Metric Learning ***Yi et al.*** (***2014***) and DeepReID ***Li et al.***

(***2014***) formulations to the sophisticated MOT methods used today ***Dong and Shen*** (***2018***); ***Liang et al.*** (***2020***); ***Wang et al.*** (***2020***); ***Yang et al.*** (***2020***).

Against this backdrop, we clarify how our formulation aligns with, and deliberately departs from, these widely adopted variants.

- **No image augmentations.** We do not apply rotations or other heavy augmentations to generate positives. Positives and negatives are instead sampled from heuristically tracked fragments: same-fragment crops provide positives, and temporally coexisting fragments provide negatives. Because crops are egocentrically pre-aligned, additional invariances are unnecessary and can suppress fine identity cues that are critical in our setting.
- **No projection head.** Unlike SimCLR-style pipelines that add a projection MLP before the contrastive loss ***Chen et al.*** (***2020a***,b), we train a lightweight encoder to output a direct, low-dimensional embedding for the sole downstream task of identity clustering. Since we do not target broad transfer or multi-task robustness, a projection head intended to filter "nuisance" features is not required.
- **No stop-gradient.** BYOL and SimSiam prevent representational collapse via stop-gradient asymmetry ***Grill et al.*** (***2020***); ***Chen and He*** (***2021***). In our case, fragments naturally yield abundant, high-quality negatives; balanced positive/negative batches and a loss-aware pair sampler maintain training signal without asymmetry. This preserves simplicity while addressing collapse risks that arise when negatives are scarce.
- **Euclidean margins rather than cosine similarity.** While many contrastive methods operate on L2-normalized features with cosine similarity, we optimize Euclidean distances with explicit thresholds D_{pos} and D_{neg}. These margins give direct control over intra- and inter-cluster spacing and align naturally with downstream K-means assignment used to form identities (as opposed to cosine-margin classifiers).
- **Novel pairs sampling strategy.** Our fragment-level, loss-aware sampling is related to online hard example mining (OHEM) and hard negative sampling for contrastive learning ***Shrivastava et al.*** (***2016***); ***Robinson et al.*** (***2021***), but it differs in two key ways that are specific to our setting. First, selection happens over *fragments* defined by temporal co-existence rather than over labeled instances or memory-bank entries, and hardness is estimated online from a decayed record of recent non-zero loss, without curated queues or global rankers. Second, the convex mixture $P = \alpha\, P_s + (1-\alpha)\, P_{l_s}$ makes the policy explicitly *exploration–exploitation*: P_s guarantees coverage of the fragment graph (preventing sampling starvation), while P_{l_s} focuses on unresolved regions of the embedding. This bandit-style view clarifies our empirical findings, pure exploitation (α=0) collapses onto a few regions (catastrophic forgetting), whereas a mid-range α maintains global progress while accelerating convergence.

Stopping criteria

Stopping network training using the loss function directly can be highly variable, as different video conditions, the number of individuals and the sampling method significantly influence this value. To circumvent this, we use the Silhouette score (SS) ***Rousseeuw*** (***1987***) of the clusters of the embedded images. Let $d(I, J)$ be the Euclidean distance between the embeddings of image I and J, for each image I, in cluster C_a we compute the mean intra-cluster distance

$$a(I) = \frac{1}{|C_a| - 1} \sum_{J \in C_a, J \neq I} d(I, J) \tag{9}$$

and the mean nearest-cluster distance

$$b(I) = \min_{a \neq b} \frac{1}{|C_b|} \sum_{J \in C_b} d(I, J) \tag{10}$$

The SS is given by

$$SS = \frac{1}{\text{number of images}} \sum_{I} \frac{b(I) - a(I)}{\max\{b(I), a(I)\}} \tag{11}$$

To determine when to stop training, every m batches we compute the SS by clustering the embeddings of a random sample of the images in the video, generating also a checkpoint of the model. m was set to be the maximum between 100 and number of animals in a video times 5. We stop training if: 1) there have been 30 consecutive SS evaluations without any improvement (patience of 30), or 2) there have been 2 consecutive SS evaluations without any improvement but the SS already achieved a value of 0.91. After stopping the training, the model with the highest SS is chosen. A threshold of 0.91 was validated empirically (***Figure 2d*** and ***Figure 2e***). The number of images used for the computation of the SS is 1000 times the number of animals.

Fallback accumulation protocol

The contrastive approach is considered to fail when it can confidently identify only less than half of the images in the video (with `CONTRASTIVE_MIN_ACCUMULATION` $= 0.5$, but users can modify this value). In such cases, and only if the video contains global fragments, protocol 2 from the original idtracker.ai is applied. This is a fallback tracking mechanism and, notably, the system never had to apply it in any of the videos of our benchmark.

The way protocol 2 from the original idtracker.ai is applied in this scenario is as follows. Fragments identified by the contrastive algorithm serve as an initial labeled dataset, effectively creating a synthetic initial global fragment in terms of the original idtracker.ai. Using this dataset, the small CNN from the original idtracker.ai is taken as the new identification network, and trained. The trained model then predicts the identities of the remaining unidentified fragments in the video. Then, quality checks are applied to filter out potentially incorrect identity assignments (see Section B.2.4, *Cascade of training/identification protocols*, in Appendix B of ***Romero-Ferrero et al.*** (***2019***)). Only fragments passing these checks are accumulated into the training dataset. The identification network is then retrained with the expanded dataset, and the prediction-checks-accumulation cycle is repeated until either (i) 99.95% of the images are assigned or (ii) no further fragments pass the quality criteria. In either case, the pipeline proceeds with the ***Residual Identification*** step.

Residual Identification

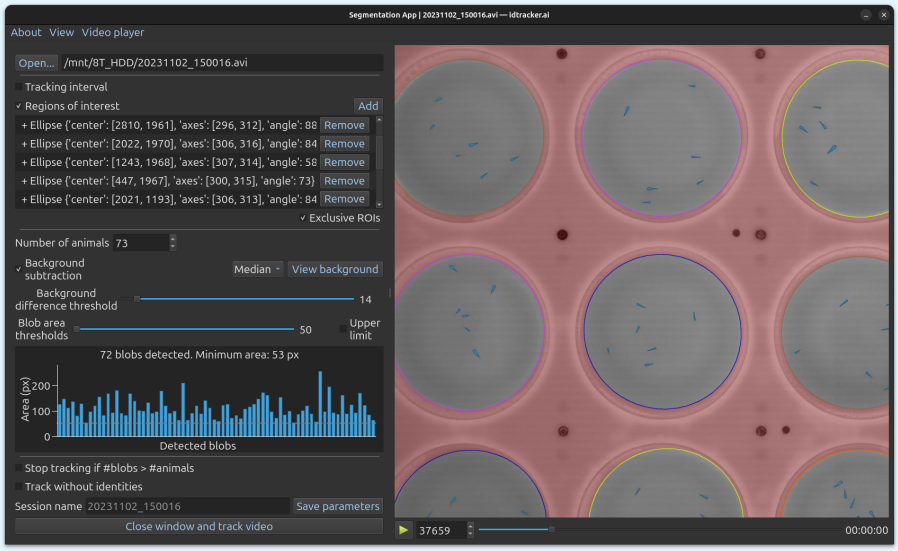
Residual identification addresses fragments that did not pass the quality checks or were excluded due to short length. The identification network (ResNet or the small CNN, depending on whether the contrastive algorithm worked well or failed, respectively), predicts the identities of these fragments and a probabilistic method that accounts for the coexistence of already assigned fragments is applied. Assignments are made iteratively in descending order of confidence, with the highest-confidence fragments resolved first (see Section B.2.5, *Residual identification*, in Appendix B of ***Romero-Ferrero et al.*** (***2019***)).

After the residual identification, the remaining processing stages are the same as in the original idtracker.ai. These include post-processing to correct impossible trajectory jumps, interpolation through crossing events, and the generation of the final output trajectories.

Appendix 4

Example Workflow

A typical workflow with idtracker.ai begins when a user records a video of a multi-animal experiment under laboratory conditions. The process starts by launching the Segmentation app (***Appendix 4—Figure 1***) using the command `idtrackerai` (see full documentation at https://idtracker.ai/latest/user_guide/segmentation_app.html). This application guides users through setting the essential tracking parameters, such as the number of animals, background subtraction options, and filters for non-animal objects (using regions of interest and minimum blob size). Once configured, tracking runs automatically. Additionally, the state of an already trained ResNet from another video can be used to initialize the current model's state using the parameter `knowledge_transfer_folder`, speeding up the training process. Users can save all these parameters to a file and execute tracking from the terminal or a script with `idtrackerai --load parameters.toml --track`, which is useful for remote or batch processing.



Appendix 4—figure 1. Segmentation GUI. Enables users to set the basic parameters required for running idtracker.ai.

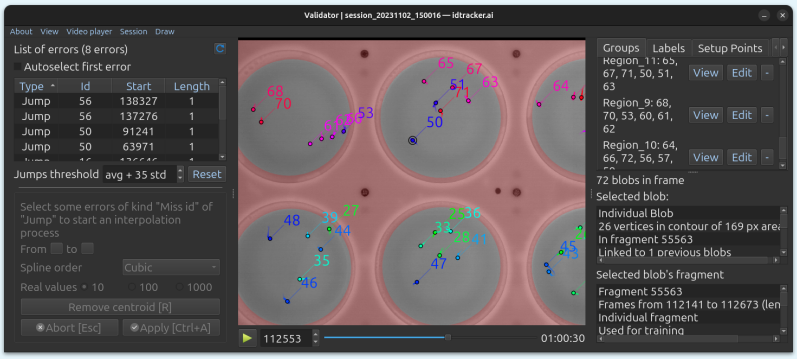
After tracking, the software outputs x-y coordinates for each identified animal in every video frame. These trajectories are saved in multiple formats (CSV, HDF5, Numpy) and include metadata such as:

- Video properties (height, width, frame rate, average blob size per identity)
- An overall estimate of tracking accuracy
- Identification probabilities for each animal in every frame
- The Silhouette score achieved during training

To assess the robustness of the learned representation space, users can generate a t-SNE plot of the embeddings from a sample of animal images by running the command `idtrackerai_inspect_clusters`, as shown in ***Figure 2c*** (see documentation at https://idtracker.ai/latest/user_guide/data_analysis.html).

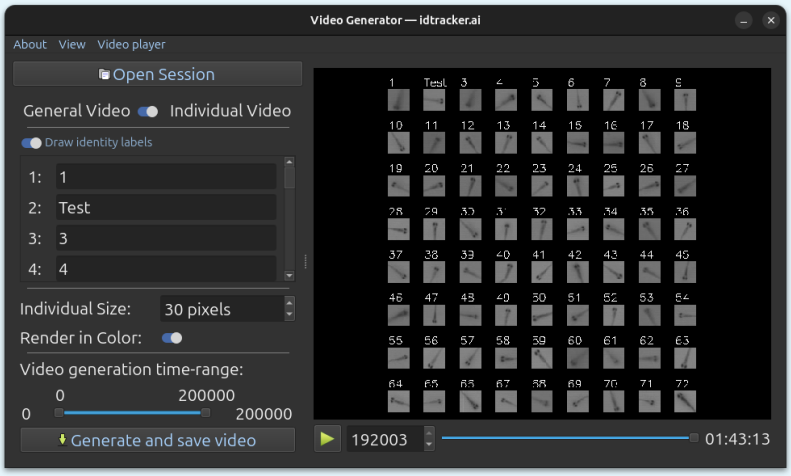
Although idtracker.ai achieves over 99% accuracy on well-recorded videos, some frames may still contain missing or mislabeled animals. To address this, users can launch the Validator app with `idtrackerai_validate` to manually review and correct tracking results (see

documentation at https://idtracker.ai/latest/user_guide/validator.html). This tool allows to navigate through video frames, inspect the tracked positions and metadata, and detect and correct errors using integrated plugins (***Appendix 4—Figure 2***).



Appendix 4—figure 2. Validator GUI. Enables users to inspect tracking results, correct errors, and access additional tools.

To share the tracking results, the command `idtrackerai_video` launches a graphical app (***Appendix 4—Figure 3***) to generate videos with animal trajectories and optional labels overlaid on the original footage (see documentation at https://idtracker.ai/latest/user_guide/video_generators.html). It can also produce individual videos for each animal, showing only its cropped region over time. These individual videos can be used as input for pose estimation tools such as DeepLabCut ***Lauer et al.*** (***2022***), SLEAP ***Pereira et al.*** (***2022***), MARS ***Segalin et al.*** (***2021***), ADPT ***Tang et al.*** (***2025***), and Lightning Pose ***Biderman et al.*** (***2024***).



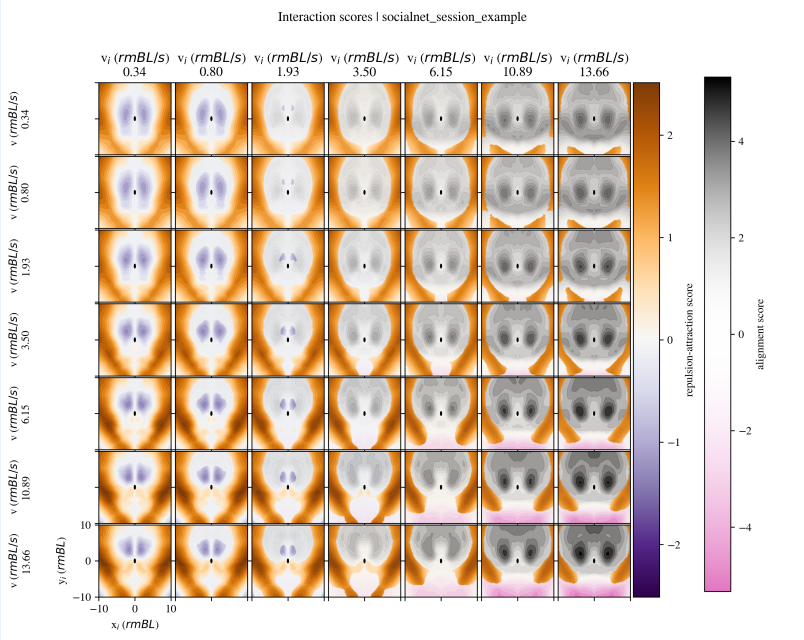
Appendix 4—figure 3. Video Generator GUI. Allows users to define parameters for general and individual video generation.

When experiments involve the same animals across multiple sessions, users can match identities between these sessions using `idmatcher.ai` (originally introduced in ***Romero-Ferrero et al.*** (***2023***) and now integrated into the idtracker.ai ecosystem; see documentation at https://idtracker.ai/latest/user_guide/idmatcherai.html and ***Appendix 5***). This tool ensures consistent identity labeling across independent recordings, supporting multi-session studies.

The resulting trajectories can be loaded and analyzed in any programming language. For this purpose, we developed the Python package *trajectorytools*. It offers utilities for the analysis of 2D trajectory data, basic movements metrics and spatial relationships between

the animals, at https://idtracker.ai/latest/user_guide/data_analysis.html#trajectorytools. In the same URL we also provide a set of Jupyter Notebooks to facilitate its usage providing analysis examples.

Additionally, SocialNet ***Heras et al.*** (***2019***), a model of collective behavior, can be seamlessly applied to the trajectories generated by idtracker.ai to extract social interaction rules, as illustrated in ***Appendix 4—Figure 4*** (see documentation at https://idtracker.ai/latest/user_guide/socialnet.html).



Appendix 4—figure 4. SocialNet output example showing learned attraction-repulsion and alignment areas for social interactions around the focal animal.

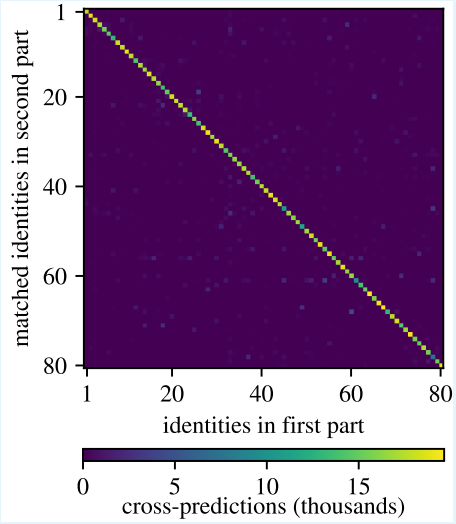
Appendix 5

Validity of idmatcher.ai in the new idtracker.ai

idmatcher.ai, first introduced in ***Romero-Ferrero et al.*** (***2023***), is an integrated tool in idtracker.ai to match individual identities across videos of the same animals. After tracking two separate videos with an arbitrary order of the identity labels, the identification networks from both are used to cross-identify animal images from the other video. This generates a confusion matrix from which final identities are obtained by solving the assignment problem with a modified Jonker-Volgenant algorithm ***Crouse*** (***2016***).

The reliability of idmatcher.ai was already proven for the networks of the previous versions of idtracker.ai (v5 and previous). To confirm its performance also in v6, several videos from the benchmark are split into two non-overlapping parts with a 200 frames gap between the them. Both parts are tracked independently and idmatcher.ai is then applied to match identities between them.

This test (with the videos *z_20*, *z_60_1*, *z_100_2*, *z_100_3*, *z_80_1*, *z_80_2*, *z_80_3*, *z_100_1*, *d_60*, *d_72*, *d_80*, *d_100_1*, *d_100_2*, and *d_100_3*) presents an average image-level accuracy of 89%, enough for a perfect identity matching with zero errors (see ***Appendix 5—Figure 1*** for an specific example with *drosophila_80*).



Appendix 5—figure 1. Confusion matrix between the two parts of *drosophila_80* in v6 of idmatcher.ai. It contains predictions both from the network trained in the first part with images from the second one and the other way around. In this example, the image-level accuracy is 82.9%, enough for a 100% accurate identity matching.

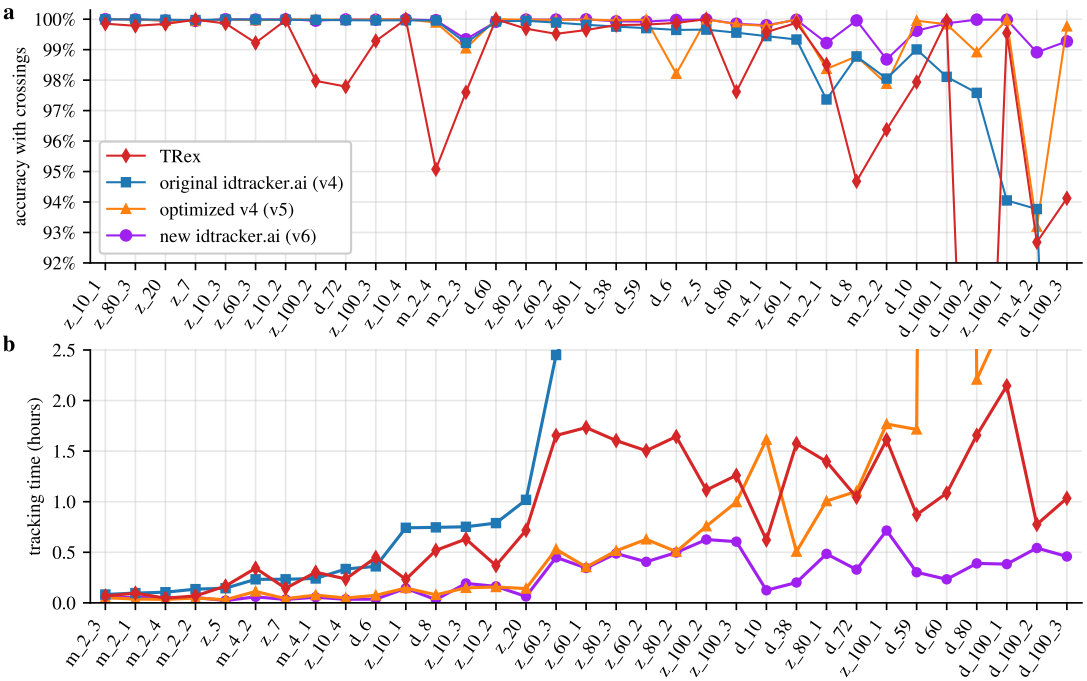


Figure 1—figure supplement 1. Performance for the benchmark with full trajectories with animal crossings. **a**. Median accuracy was computed using all images of animals in the videos including animal crossings. **b**. Median tracking times. ***Supplementary Table 1***, ***Supplementary Table 2***, ***Supplementary Table 3*** and ***Supplementary Table 4*** give more complete statistics (median, mean and 20-80 percentiles) for the original idtracker.ai (version 4 of the software), optimized v4 (version 5), new idtracker.ai (version 6) and TRex, respectively.

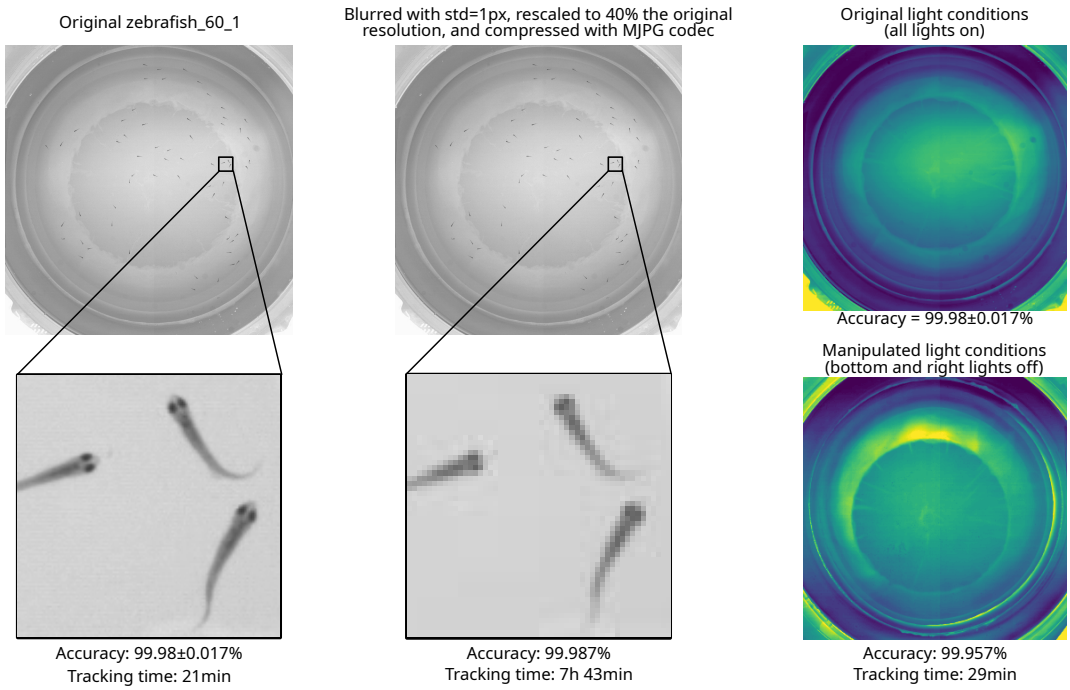


Figure 1—figure supplement 2. Robustness to blurring and light conditions. First column: Unmodified video *zebrafish_60_1*. **Second column:** *zebrafish_60_1* with a gaussian blurring of 1 pixel plus a resolution reduction to 40% of the original plus MJPG video compression. Versions 5 of idtracker.ai fails to track this videos with a 82% accuracy. **Third column:** Videos of 60 zebrafish with manipulated light conditions (same test as in idtracker.ai ***Romero-Ferrero et al.*** (***2019***)). **First row:** Uniform light conditions across the arena (*zebrafish_60_1*). **Second row:** Similar setup but with lights off in the bottom and right side of the arena.

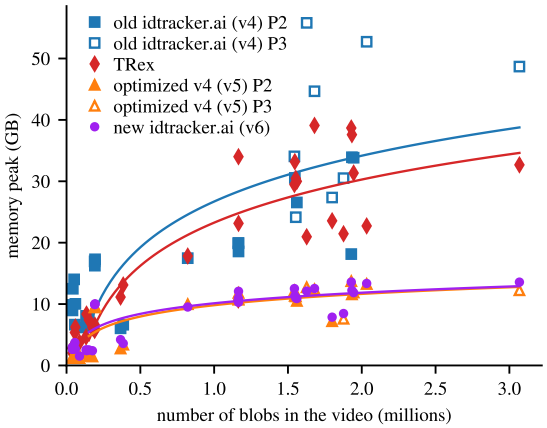


Figure 1—figure supplement 3. Memory usage across the different softwares. The solid line is a logarithmic fit to the (RAM) memory peak as a function of the number of blobs in a video. The Python tool *psutil.virtual_memory().used* was used to measure the memory usage 5 times per second during the tracking. The presented value is the maximum measured value minus the baseline measured before the tracking start. Disclaimer: Both softwares include automatic optimizations that adjust based on machine resources, so results may vary on systems with less available memory. These results were measured on computers with the specifications in ***Methods***.

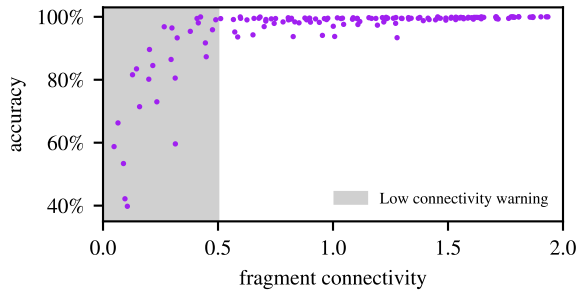


Figure 3—figure supplement 1. Fragment connectivity analysis. The relation between fragment connectivity and the corresponding accuracy for every video and angle θ of the new idtracker.ai in ***Figure 3***. Fragment connectivity is calculated as the average number of other fragments each fragment coexists with, divided by $N-1$, with N the total number of animals. Values below 0.5 are associated with low accuracy, and idtracker.ai alerts the user in such cases.