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 440 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.

While both idTracker and idtracker.ai perform well in high-quality video, they share a limitation that can be critical in videos of lower quality or with many occlusions. To understand this limitation, consider the schematics of a video in ***Figure 1a***. The first step of both idTracker and idtracker.ai consists in detecting instances when animals touch or cross paths (***Figure 1a***, shown as boxes with dashed borders and containing images of overlapping fish in this example). The video is then divided into individual fragments, each consisting of the set of images of a single individual between two animal crossings (***Figure 1a*** shows 14 of them as rectangles with a gray background). A global fragment for a video with N animals is a collection of N fragments that coexist in one or more consecutive frames in the video (***Figure 1a***, the 5 fragments with blue borders are a global fragment). The significance of a global fragment is that it provides a set of images and identity labels for all the animals in the video.

The core idea of idTracker and the original idtracker.ai is to use global fragments for the classification of images of animals into identities. In idtracker.ai, this process starts by training a convolutional neural network (CNN) with the images and labels of the global fragment that contains the longest fragment for the animal that moves the least. Once trained, the network assigns identities to all animal images in the remaining global fragments. Only global fragments meeting strict quality criteria, such as ensuring all animals in a global fragment have unique identities, are retained for further training. This iterative process of training, assigning, and selecting continues until most of the video have images assigned to identities. A second algorithm then tracks animals during crossings given that animals are already identified outside crossings.

Figure 2a (blue line) shows the accuracies of the original idtracker.ai (version 4 of the software) for a benchmark of 33 videos of zebrafish, flies and mice. These accuracies were computed using all the images of animals in the videos excluding animal crossings. ***Figure 2—figure Supplement 1a*** shows the same results but for the complete trajectory with animal crossings. The names of the videos start with a letter for the species (z,f,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. The videos in this figure are ordered by decreasing accuracy of the original idtracker.ai results for ease of visualization. The first 15 videos are videos of zebrafish, flies and mice with an accuracy of $> 99.9\%$. The accuracy in the remaining videos gradually decreases to 92.67% in video m_4_2, and a value of 50.4% outside the figure for video d_100_3.

Figure 2b (blue line) shows the times that the original idtracker.ai takes to track each of the videos in the benchmark. The videos are ordered by increasing tracking times for ease of visualization. The original idtracker.ai has a faster protocol, "Protocol 2", which works well for the simplest videos and its tracking times ranging from a few minutes to several hours. However, for complex videos, the software may switch from "Protocol 2" to "Protocol 3", with Protocol 3 a two-step process. In the first step, all the global fragments are used to train the CNN filters. The second step proceeds like Protocol 2 but with the initial weights of the CNN filters obtained from the first step. While effective, this approach can be extremely slow, often requiring several days or weeks for a single video. Since it is stochastic whether a video is tracked using Protocol 2 or 3 (***Figure 2—figure Supplement 2***), a reasonable strategy to use the original idtracker.ai is to track each video multiple times until Protocol 2 successfully tracks the entire video or, when a patience threshold is reached (here set to 5 attempts), switch to Protocol 3. The tracking times shown in ***Figure 2b*** (blue line) correspond to this procedure, with the time being the accumulated time of the multiple attempts made by the software until final tracking. 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. If we were to run idtracker.ai a single time instead of following this protocol, the tracking times for some of the videos would be longer.

We first optimized idtracker.ai by improving data loading protocols and redefining the main objects in the software (animal images and fragments) and their properties (see ***Methods*** for details). This version of the optimized original idtracker.ai (version 5 of the software) achieved better accuracies, ***Figure 2a*** (orange line), and ***Figure 2—figure Supplement 1a*** (orange line) for accuracies including animal crossings. The mean accuracy across the benchmark for this optimized version is 99.58% and 99.40% including or not animal crossings, respectively, while for the original idtracker.ai are 97.52% and 97.38%.

Even if this version also uses Protocols 2 and 3, we obtain much shorter tracking times, never longer than a day ***Figure 2b*** (orange line). On average, tracking is 13.6 times faster than with the original idtracker.ai and, for the more difficult videos, 118.4 times faster. However, waiting a day to track some videos can make a tracking pipeline too slow. To further improve accuracy and tracking times, we retained these optimizations while also changing the main logic of idtracker.ai. In the original idtracker.ai, when global fragments are short, the quality of the initial CNN is low, leading to either reduced accuracy or the triggering of the very slow Protocol 3. The new system had to be able to track without global fragments.

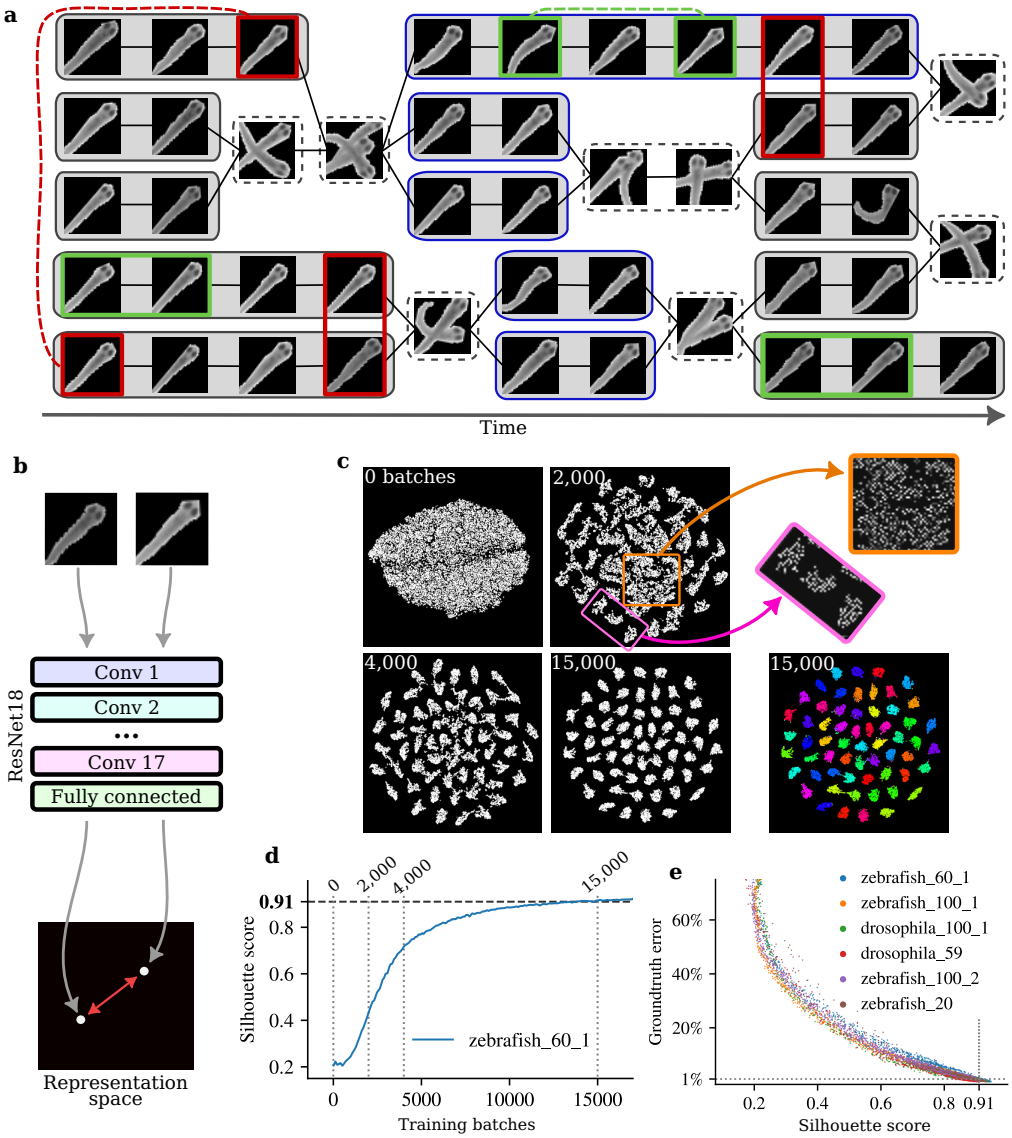

Figure 1. Tracking by identification using deep contrastive learning. **a** Schematic representation of a video with five fish. It shows 7 portions of video with animals crossing or touching (dashed-border boxes), and 14 individual fragments, sequences of images of a single individual between two crossings (gray-background boxes). The blue-border fragments form a global fragment, as there are as many individual fragments as animals and all the individual fragments coexist in one or more frames. Some pairs of images of the same animal identity are highlighted with green borders (positive images) and some images of different identities are highlighted with red borders (negative images). **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. 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.

Figure 1—figure supplement 1. Models comparison

Figure 1—figure supplement 2. Embedding dimensions comparison

Figure 1—figure supplement 3. D_{neg} over D_{pos} comparison

Figure 1—figure supplement 4. Batch size comparison

Figure 1—figure supplement 5. Exploration and exploitation comparison

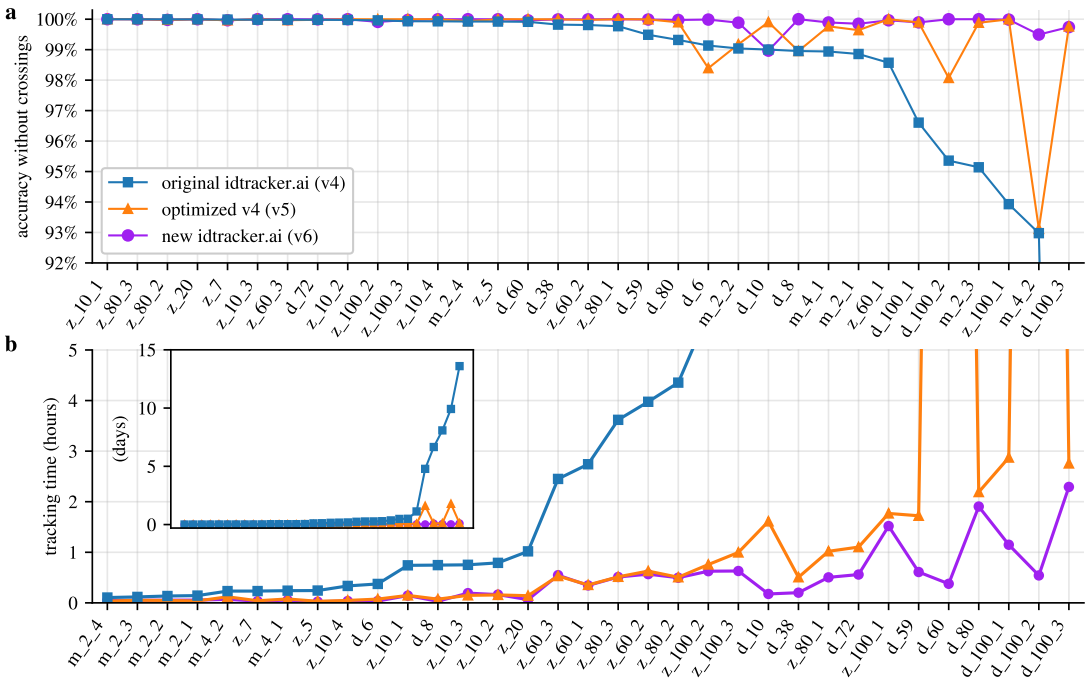

Figure 2. Performance for a benchmark of 33 videos of flies, zebrafish and mice. **a**. Median accuracy was computed using all images of animals in the videos excluding animal crossings. **b**. Median tracking times are shown for the scale of hours and, in the inset, for the scale of days. ***Supplementary 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.

Figure 2—figure supplement 1. Performance for the benchmark with full trajectories with animal crossings

Figure 2—figure supplement 2. Protocol 2 failure rate

Figure 2—figure supplement 3. Memory usage across the different softwares.

Figure 2—figure supplement 4. Robustness to blurring and light conditions

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 clustering into animal identities without needing identity labels. This is possible due to the structure of the video, ***Figure 1a***. Note that pairs of images of the same individual can be obtained from the same fragment (***Figure 1a***, green boxes). Also, pairs of images from different individuals can be obtained from different fragments that coexist in time for one or more frames (***Figure 1a***, red boxes). These pairs can be used as "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***).

We first evaluated neural networks suitable for contrastive learning with animal images. In addition to our previous CNN from idtracker.ai, we tested 23 networks from 8 different families of state-of-the-art convolutional neural network 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 ***He et al.*** (***2016***) was the fastest to obtain low errors (***Figure 1—figure Supplement 1***).

A ResNet18 with M outputs maps each input image to a point in an M-dimensional representation space (illustrated in ***Figure 1b*** as a point on a plane). Experiments showed that using $M = 8$ achieved faster convergence to low error (***Figure 1—figure Supplement 2***). ResNet18 is trained us-

ing a contrastive loss function (***Chopra et al.*** (***2005***), see ***Methods*** for details). Each image in a positive or negative pair is input separately into the network, producing a point in the 8-dimensional representation space. For an image pair, we then obtain two points in an 8-dimensional space, separated by some (Euclidean) distance. The loss function is used to minimize (or maximize) this Euclidean distance for positive (or negative) pairs until the distance D_{pos} (or D_{neg}). 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 (***Figure 1—figure Supplement 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 1c*** 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 1c***, 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 1c*** shows at $2,000$ training batches that some clusters well-defined (e.g. those inside the orange square) while others remain fuzzy (e.g. those inside the pink rectangle). 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 (***Methods***). 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 index ***Rousseeuw*** (***1987***), which quantifies intra-cluster cohesion and inter-cluster separation. A maximum value of 1 indicates ideal clustering. During training, the mean Silhouette index increases (***Figure 1d***). 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 1e***). As a result, we use 0.91 as the stopping criterion for training (***Methods***).

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 has a higher accuracy than original idtracker.ai and than its optimized version, ***Figure 2a*** (magenta line). Its average accuracy in the benchmark is 99.92% and 99.78% without and with crossings, respectively, an important improvement over the original idtracker.ai (97.52% and 97.38%) and its optimized version (99.58% and 99.40%). It also gives much shorter times than the original idtracker.ai and its optimized version, ***Figure 2b*** (magenta line). It is on average 44 times faster than the original idtracker.ai and, for the more difficult videos, up to 440 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 2—figure Supplement 4***). We also compared the new idtracker.ai to TRex ***Walter and Couzin*** (***2021***), which is based on Protocol 2 of idtracker.ai but with additional operations like eroding crossings to make global fragments longer.

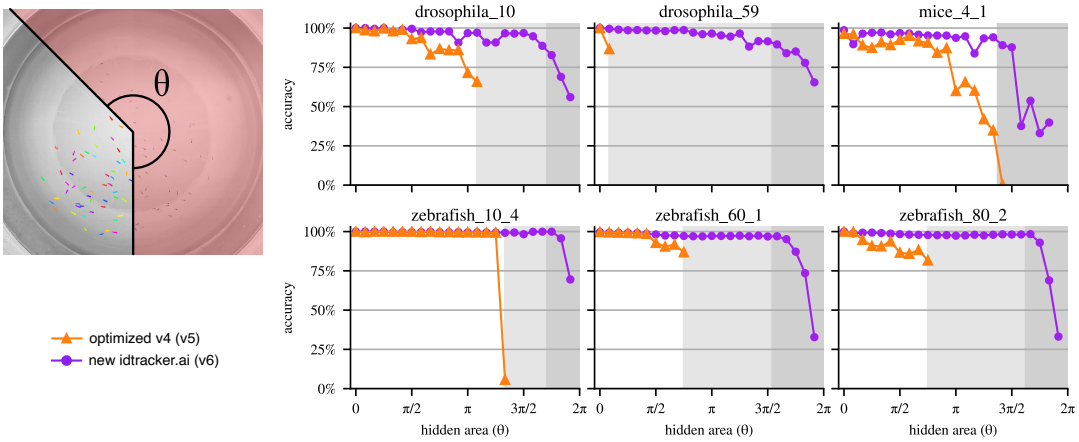

Figure 3. Tracking with strong occlusions. Accuracies when we mask a region of a video defined by an angle θ and the tracking system has no access to the information behind the mask. Light and dark gray region correspond to the angles for which no global fragments exist in the video. Dark gray regions correspond to angles for which the video does not have enough coexisting individual fragments, specifically on average less than $0.25(N-1)$ coexisting individual fragments, with N the number of animals in the video. 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.

TRex gives comparable accuracies to the original idtracker.ai in the benchmark, but by avoiding Protocol 3, it is on average 31 times faster than the original idtracker.ai and up to 315 times faster (***Figure 2—figure Supplement 1b***). However, the new idtracker.ai is both more accurate and faster than TRex (***Figure 2—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 99.78%, respectively. Also, the new idtracker.ai is on average 3.9 times faster and up to 16.5 times faster than TRex. Additionally, the new idtracker.ai has a memory peak lower than TRex (***Figure 2—figure Supplement 3***).

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. To study this systematically, we added a mask on the video with an angle θ (***Figure 3***). The tracking systems have no access to the information behind the mask. 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. The new idtracker.ai, however, works well until approximately $1/4$ of the setup is visible, and afterward it degrades. This also shows the limit of the new idtracker.ai. For the clustering process to be successful, we need enough coexisting individual fragments to have both positive and negative examples. Empirically, we find a deterioration with less than $0.25(N-1)$ coexisting individual fragments, with N the number of animals in the video (***Figure 3***, dark gray region). The new idtracker.ai flags when this condition is not met.

The final output of the new idtracker.ai consists of the $x-y$ coordinates for each identified animal and video frame. Additionally, it provides several quality metrics: an estimate of the probability of correct identity assignment for each animal and frame, the Silhouette score as a measure of clustering quality, and the average number of coexisting individual fragments per fragment divided by $(N-1)$, with N the number of animal in the video, which when above $0.25(N-1)$ is expected to give good results. The software can also generate a video with the computed animal trajectories for visualization, and an individual video per animal 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***). For analysis of trajectories and spatial relationships, the user can run our Python package

trajectorytools on the trajectories.

In summary, the new idtracker.ai takes an approach to tracking using representational learning to avoid the need for segments of the video in which all animals are visible. This makes the new idtracker.ai work in more videos, more accurately, much faster and with a lower memory peak.

Acknowledgments

We thank Alfonso Perez-Escudero, Paco Romero-Ferrero, Francisco J. Hernandez Heras, and Madalena Valente for discussions. 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, T.C. wrote the Appendices with help from J.T and 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 GPL v.3). Information about its installation and usage can be found on the official website https://idtracker.ai. The source code is available in 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 v6.0.0.

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, and Windows 11 with NVIDIA GPUs from the 1000 to the 4000 series and MacOS 15 with Metal chips. The benchmark presented in this study was performed on 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.

Improvements to 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

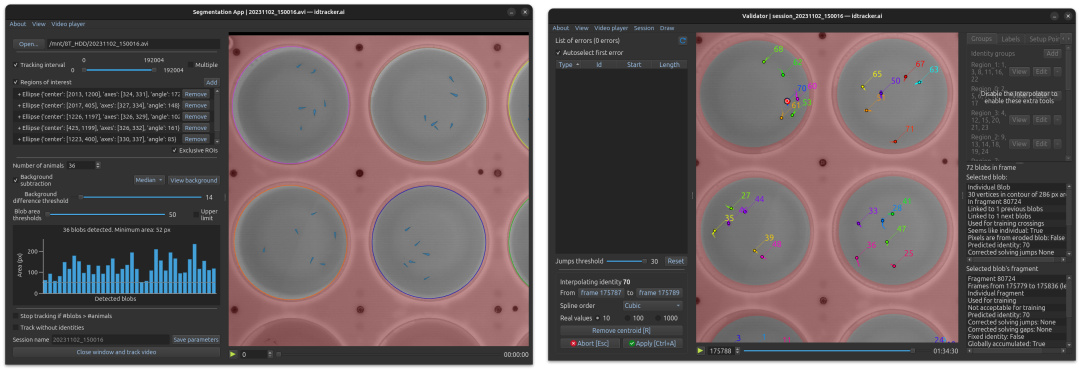

Figure 4. idtracker.ai new graphic user interface. New graphics user interface (GUI) for versions v5 and v6 of idtracker.ai. On the left the segmentation GUI. On the right the Validator tool.

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.

Improved graphical user interface and introduction of Exclusive ROIs: The graphical user interface was redesigned for improved usability and now includes the "Exclusive Regions of Interest" feature, which allows users to define spatially distinct regions in multi-arena experiments where animal identities are treated independently (see ***Figure 4*** left image). It also incorporates a redesigned video generator for visualizing tracking results.

Validation application: A standalone GUI for inspecting and correcting tracking results. It allows users to navigate video frames, review tracked positions and metadata, detect tracking errors, and apply corrections using integrated plugins (see ***Figure 4***, right image).

Direct integration with `idmatcher.ai`: A utility for matching identities across videos, originally introduced in ***Romero-Ferrero et al.*** (***2023***). It allows users to propagate consistent identity labels across multiple recordings, facilitating longitudinal or multi-session experiments. It is now a native feature of both v5 and v6, fully integrated into the idtracker.ai ecosystem.

Protocol details for the new idtracker.ai

In this section, we give an overview of the tracking protocol. Please refer to **Appendix 1** for details.

Architectures

The *contrastive learning* network (***Figure 1b***) is a ResNet18 ***He et al.*** (***2016***) with a single channel in the first convolutional layer for grayscale images and 8 neurons in the last layer. The network receives grayscale images because idtracker.ai always works with grayscale converted video frames.

Loss function

The contrastive loss L for a pair of images (I, J) and label l is defined as:

$$\mathcal{L}\left(I, J, l\right) = l_{I,J} \cdot \max(0, D_{I,J} - D_{\text{pos}})^2 + (1 - l) \cdot \max(0, D_{\text{neg}} - D_{I,J})^2$$

$$l = \begin{cases} 1 & \text{if I and J come from the same fragment, (positive pair)} \\ 0 & \text{if I and J come from coexisting fragments (negative pair)} \end{cases}$$

Here $D_{I,J}$ is the Euclidean distance between the embeddings of images I and J. D_{pos} is the maximum allowed distance between the two images of a positive pair, and D_{neg}, the minimum allowed distance between the two images in negative pair.

Training

ResNet18 is trained using Adam optimizer with the hyperparameters described in ***Kingma and Ba*** (***2017***). The learning rate is set at the value of 0.001 using training batches of 1600 images (400 positive pairs and 400 negative pairs of images). See **Appendix 2** for details.

Pair selection

The selection of pairs was done by combining two sampling strategies:

1. **Sampling fragments according to their size** so that fragments containing more images are sampled more often.
2. **Sampling fragments according to the loss function** by increasing the sampling probability of pairs of fragments from whom the corresponding images had positive loss, and decreasing the sampling probability of pairs of fragments from whom the corresponding images had loss zero.

See **Appendix 2** for more details on the pair sampling strategy.

Clustering and stopping criteria

For clustering, we use the minibatch K-means clustering, which significantly reduces the computation time compared to a classical implementation ***Sculley*** (***2010***).

Stopping of the training was done by computing the K-means clustering for a subset of (number of animals times 1,000) images, and measuring the corresponding Silhouette score (SS) ***Rousseeuw*** (***1987***) every number of animals times 5 batches. We stop training if there have been 30 consecutive SS evaluations without any improvement (patience of 30), or if there have been 2 consecutive SS evaluations without any improvement but the SS already achieved the target value 0.91. Check **Appendix 2** for more details on the criteria to stop the training of the network.

Occlusion tests

For the occlusion tests, we took videos of freely behaving animals in a round arena (included in the benchmark) and occluded a sector of the circle between 0 and θ radians. For the tracking software, animals disappeared when entering this occluded section of the arena. The light gray area in ***Figure 3*** corresponds to a degree of occlusion that prevents the existence of global fragments. The

dark gray area in ***Figure 3*** corresponds to a degree of occlusion where there are less than $0.25(N-1)$ coexisting individual fragments (N being the number of animals in the video). With these degrees of occlusion, too few animals overlap at any given time and identification is expected to deteriorate in this regime (***Figure 3***, dark gray region). idtracker.ai flags when this condition is not met.

Computation of tracking accuracy

Using the idtracker.ai Validator tool (see ***Methods***), we manually generated ground-truth trajectories based on v5 outputs. This ground-truth consists on the positions and identities of all animals in each frame and their classification as either individual or crossing.

To detect tracking errors, we analyze the video frame by frame, verifying whether the predicted position of each animal deviates from the ground-truth by more than a threshold T. Errors are also recorded when the software loses the identity or fails to detect an animal in a given frame.

Tracking accuracy is then defined as one minus the proportion of errors in the trajectory. For **accuracy with crossings**, we consider all trajectory points, whereas for **accuracy without crossings**, we exclude points corresponding to crossing events in the ground-truth.

We present all results using a threshold $T = 1$BL with BL being a body length. We also verified that accuracy remains largely unaffected by the value of this threshold. 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 99.92% to 99.90%.

Benchmark of accuracy and tracking time

To evaluate the tracking time and accuracy of different versions 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 and features one of three species: mice, drosophila, or zebrafish, with the number of individuals ranging from 2 to 100 (see ***Methods***).

Previous versions of idtracker.ai (v4 and v5) can resort to protocol 3 for tracking, a method that can take days to process more complex videos but is necessary when protocol 2 fails. Similarly, TRex, lacking an equivalent of protocol 3, can fail to track certain videos, leading to missing accuracy outputs (***Figure 2—figure Supplement 2***).

To estimate the accuracy and tracking time that a standard user might experience, we simulate a realistic user workflow. This simulation accounts for the possibility that the software may fail to track the video, prompting the user to try again with a slightly different parameter configuration, up to a certain number of attempts.

The user is given up to 5 attempts to successfully track a video. Attempts are sampled from a precomputed dataset of tracking runs. Accuracy is taken from the first successful run. The reported tracking time is the sum of the time taken by that successful run and all preceding failed attempts. In cases where all attempts fail, accuracy is determined by protocol 3 (in v4 and v5 of idtracker.ai), and tracking time includes the time required for protocol 3 plus the total time of all failed attempts. This sampling process is repeated 10,000 times per software and video to obtain statistically robust estimates of the tracking times and accuracies. ***Figure 2*** and ***Figure 2—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.

Dataset of tracking runs

To build the dataset of tracking runs we used for the benchmark of accuracies and times, we define input parameters through each software's graphical interface. Fixed parameters (e.g., number of animals, regions of interest) are held constant, while those with multiple valid values are treated as variable, with their ranges annotated. In idtracker.ai, the variable parameter is the `intensity_threshold`, whereas in TRex, the variable parameters are `threshold` and `track_max_speed`.

Tracking is repeated for each video and software until either 5 successful runs or 35 total runs are reached. For the original version of idtracker.ai, this is limited to 3 successful runs or 7 total runs due to significantly longer tracking times. In successful runs, both accuracy and tracking time are recorded. In failed runs, when idtracker.ai defaults to protocol 3 or TRex fails to output identities (see ***Figure 2—figure Supplement 2***), only the time until failure is recorded. For previous idtracker.ai versions (v4 and v5), failure time corresponds to the time until the software switched to protocol 3.

Each tracking run is conducted by randomly sampling values for the variable parameters from the annotated ranges and executing the full tracking process. To ensure a fair comparison, TGrabs is included when running TRex, graphical interfaces are always disabled at runtime to maximize performance, and `output_interpolate_positions` is enabled in TRex.

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.

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. .

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. .

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.

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.

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.

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, 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.

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.

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.

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.

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.

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.

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.

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.

Supplementary tables

	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	99.135	99.135	98.536 - 99.734	99.055	99.055	98.514 - 99.596	0:22:27	0:22:27	0:21:39 - 0:23:15
drosophila_8	98.955	98.663	98.369 - 98.955	98.603	98.310	98.015 - 98.603	0:44:47	0:48:50	0:30:51 - 1:15:10
drosophila_10	99.003	99.003	one point only	99.004	99.004	one point only	6:28:25	6:28:25	6:18:34 - 6:38:12
drosophila_38	99.822	99.756	99.688 - 99.822	99.699	99.618	99.535 - 99.699	6:55:15	6:15:08	4:16:03 - 7:23:33
drosophila_59	99.489	99.489	99.489 - 99.489	99.441	99.441	99.441 - 99.441	1 day, 3h	1 day, 3h	9:41:00 - 1 day, 20h
drosophila_60	99.914	99.914	one point only	99.848	99.848	one point only	4 days, 18h	4 days, 18h	4 days, 16h - 4 days, 21h
drosophila_72	99.980	99.965	99.949 - 99.980	99.960	99.948	99.934 - 99.960	11:35:20	13:00:53	7:00:26 - 16:42:43
drosophila_80	99.319	99.319	one point only	99.220	99.220	one point only	6 days, 15h	6 days, 15h	6 days, 15h - 6 days, 16h
drosophila_100_1	96.605	96.605	one point only	96.344	96.344	one point only	8 days, 1h	8 days, 1h	8 days, 1h - 8 days, 2h
drosophila_100_2	95.358	95.358	one point only	95.314	95.314	one point only	9 days, 22h	9 days, 22h	9 days, 21h - 9 days, 22h
drosophila_100_3	54.021	54.021	one point only	53.758	53.758	one point only	13 days, 14h	13 days, 14h	13 days, 13h - 13 days, 15h
mice_2_1	98.858	98.851	98.845 - 98.858	97.646	97.328	97.006 - 97.646	0:08:36	0:07:12	0:05:47 - 0:08:36
mice_2_2	99.039	98.980	98.919 - 99.039	97.998	97.999	97.998 - 98.000	0:08:08	0:07:19	0:06:30 - 0:08:08
mice_2_3	95.140	97.528	95.140 - 99.953	94.695	96.737	94.695 - 98.810	0:07:03	0:06:01	0:04:58 - 0:07:03
mice_2_4	99.924	99.947	99.924 - 99.971	99.919	99.942	99.919 - 99.966	0:06:19	0:05:24	0:04:28 - 0:06:19
mice_4_1	98.940	98.711	98.480 - 98.940	98.944	98.613	98.278 - 98.944	0:14:27	0:12:37	0:10:45 - 0:14:27
mice_4_2	92.977	90.780	88.553 - 92.977	93.046	90.865	88.654 - 93.046	0:13:58	0:14:56	0:13:58 - 0:15:55
zebrafish_5	99.922	99.652	99.375 - 99.922	99.910	99.617	99.317 - 99.910	0:14:36	0:11:40	0:08:40 - 0:14:36
zebrafish_7	99.987	99.976	99.965 - 99.987	99.946	99.939	99.932 - 99.946	0:13:59	0:15:38	0:13:59 - 0:17:22
zebrafish_10_1	99.999	99.999	99.999 - 99.999	99.994	99.996	99.994 - 99.998	0:44:31	0:44:12	0:42:42 - 0:45:24
zebrafish_10_2	99.975	99.975	99.975 - 99.976	99.953	99.959	99.953 - 99.965	0:47:27	0:47:23	0:47:18 - 0:47:27
zebrafish_10_3	99.983	99.984	99.983 - 99.984	99.982	99.976	99.971 - 99.982	0:45:08	0:44:05	0:43:02 - 0:45:08
zebrafish_10_4	99.930	99.955	99.930 - 99.980	99.929	99.954	99.929 - 99.979	0:20:05	0:19:19	0:18:32 - 0:20:05
zebrafish_20	99.994	99.996	99.994 - 99.997	99.963	99.960	99.957 - 99.963	1:01:06	0:56:15	0:51:24 - 1:01:06
zebrafish_60_1	98.571	98.622	98.571 - 98.673	98.575	98.626	98.575 - 98.676	2:44:27	2:54:04	2:44:27 - 3:03:42
zebrafish_60_2	99.809	99.881	99.809 - 99.955	99.783	99.857	99.783 - 99.934	3:58:33	3:02:13	2:04:41 - 3:58:33
zebrafish_60_3	99.982	99.980	99.979 - 99.982	99.976	99.976	99.975 - 99.976	2:27:01	2:34:41	2:27:01 - 2:42:29
zebrafish_80_1	99.770	99.848	99.703 - 99.997	99.720	99.818	99.661 - 99.983	8:34:31	12:25:34	6:29:28 - 11:47:52
zebrafish_80_2	99.995	99.949	99.901 - 99.995	99.988	99.943	99.896 - 99.988	4:21:16	4:19:10	4:17:00 - 4:21:16
zebrafish_80_3	99.998	99.946	99.894 - 99.998	99.983	99.933	99.882 - 99.983	3:37:08	4:21:29	3:37:08 - 5:05:52
zebrafish_100_1	93.929	96.881	93.929 - 99.862	93.467	96.631	93.467 - 99.825	11:55:32	12:03:34	10:12:42 - 13:06:53
zebrafish_100_2	99.962	99.965	99.962 - 99.969	99.953	99.955	99.953 - 99.958	5:44:14	5:35:00	5:25:35 - 5:44:14
zebrafish_100_3	99.933	99.906	99.880 - 99.933	99.922	99.897	99.870 - 99.922	6:20:55	6:01:34	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	98.392	98.238	97.285 - 98.402	98.111	98.109	97.331 - 98.127	0:04:28	0:04:37	0:03:37 - 0:05:13
drosophila_8	98.955	99.165	98.953 - 98.999	98.603	98.861	98.585 - 98.688	0:04:23	0:04:59	0:03:39 - 0:05:14
drosophila_10	99.903	98.618	96.103 - 99.903	99.903	98.618	96.103 - 99.903	1:36:38	1:15:26	0:30:41 - 1:42:51
drosophila_38	99.994	99.974	99.987 - 99.995	99.932	99.909	99.916 - 99.952	0:30:22	0:31:23	0:23:22 - 0:37:52
drosophila_59	99.994	99.867	99.724 - 100.000	99.971	99.855	99.716 - 99.995	1:43:19	2:01:38	1:12:12 - 2:29:09
drosophila_60	100.000	99.932	99.774 - 100.000	100.000	99.908	99.654 - 100.000	1 day, 14h	1 day, 5h	3:51:31 - 1 day, 15h
drosophila_72	99.980	99.985	99.979 - 99.993	99.964	99.969	99.961 - 99.980	1:06:10	1:23:35	0:46:16 - 1:42:32
drosophila_80	99.897	99.904	99.877 - 99.925	99.726	99.724	99.715 - 99.741	2:11:25	4:56:26	1:24:11 - 3:23:58
drosophila_100_1	99.895	99.830	99.647 - 99.945	99.723	99.659	99.492 - 99.749	2:52:21	10:45:30	1:45:07 - 1 day, 2h
drosophila_100_2	98.070	98.070	one point only	98.030	98.030	one point only	1 day, 18h	1 day, 18h	1 day, 17h - 1 day, 19h
drosophila_100_3	99.760	99.735	99.599 - 99.770	99.641	99.613	99.471 - 99.645	2:45:22	8:04:32	1:39:37 - 4:11:32
mice_2_1	99.640	99.639	99.579 - 99.724	98.250	98.225	97.735 - 98.541	0:02:15	0:02:21	0:02:14 - 0:02:29
mice_2_2	99.176	99.162	99.053 - 99.246	97.881	97.970	97.687 - 98.493	0:02:50	0:02:48	0:02:28 - 0:03:01
mice_2_3	99.883	98.795	94.339 - 100.000	98.633	97.669	93.397 - 99.124	0:03:05	0:03:08	0:02:54 - 0:03:39
mice_2_4	99.937	99.935	99.863 - 100.000	99.871	99.868	99.828 - 99.904	0:02:04	0:02:07	0:02:01 - 0:02:25
mice_4_1	99.765	99.593	99.291 - 99.969	99.640	99.324	98.873 - 99.756	0:04:36	0:04:58	0:04:34 - 0:06:53
mice_4_2	93.117	92.985	92.294 - 93.128	93.058	92.906	92.287 - 93.087	0:07:12	0:08:38	0:04:37 - 0:12:06
zebrafish_5	99.998	99.997	99.998 - 99.999	99.984	99.984	99.983 - 99.984	0:01:51	0:01:49	0:01:40 - 0:02:03
zebrafish_7	99.963	99.539	98.800 - 99.967	99.909	99.505	98.776 - 99.938	0:02:29	0:03:11	0:02:23 - 0:05:05
zebrafish_10_1	100.000	100.000	100.000 - 100.000	100.000	100.000	99.999 - 100.000	0:08:50	0:08:47	0:07:53 - 0:09:30
zebrafish_10_2	99.999	99.952	99.763 - 100.000	99.989	99.941	99.747 - 99.991	0:09:24	0:09:51	0:09:21 - 0:11:48
zebrafish_10_3	100.000	99.713	99.999 - 100.000	99.994	99.712	99.993 - 99.996	0:08:54	0:09:15	0:08:32 - 0:08:54
zebrafish_10_4	99.999	99.996	99.986 - 99.999	99.997	99.994	99.984 - 99.999	0:02:52	0:02:55	0:02:44 - 0:02:58
zebrafish_20	99.997	99.992	99.992 - 99.999	99.901	99.906	99.898 - 99.932	0:08:29	0:08:38	0:07:42 - 0:10:46
zebrafish_60_1	99.999	99.999	99.997 - 100.000	99.994	99.994	99.992 - 99.994	0:21:14	0:21:33	0:20:34 - 0:24:17
zebrafish_60_2	99.978	99.894	99.947 - 99.999	99.957	99.878	99.924 - 99.992	0:37:40	0:33:28	0:20:44 - 0:43:23
zebrafish_60_3	99.967	99.965	99.924 - 99.998	99.929	99.934	99.888 - 99.960	0:31:47	0:35:43	0:30:53 - 0:43:33
zebrafish_80_1	99.999	99.993	99.999 - 99.999	99.982	99.977	99.982 - 99.984	1:01:10	1:27:36	0:47:00 - 1:22:16
zebrafish_80_2	99.978	99.970	99.922 - 99.998	99.970	99.962	99.915 - 99.989	0:30:23	0:29:19	0:27:24 - 0:30:27
zebrafish_80_3	100.000	100.000	100.000 - 100.000	99.985	99.985	99.978 - 99.989	0:30:52	0:54:43	0:27:00 - 1:29:25
zebrafish_100_1	99.996	99.988	99.966 - 99.996	99.956	99.953	99.938 - 99.958	1:46:04	1:48:59	1:30:47 - 2:09:56
zebrafish_100_2	99.998	99.976	99.909 - 99.998	99.985	99.967	99.902 - 99.990	0:45:27	0:46:22	0:36:17 - 0:50:22
zebrafish_100_3	99.999	99.984	99.999 - 100.000	99.989	99.974	99.986 - 99.991	0:59:47	1:03:35	0:33:26 - 1:06:14

Supplementary Table 2. Performance of optimized v4 (v5) 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	99.988	99.914	99.976 - 99.994	99.950	99.871	99.941 - 99.972	0:02:12	0:02:15	0:02:10 - 0:02:17
drosophila_8	100.000	99.999	99.995 - 100.000	99.922	99.864	99.638 - 99.927	0:01:53	0:01:50	0:01:39 - 0:01:54
drosophila_10	98.968	99.167	98.968 - 98.969	98.969	99.168	98.969 - 98.970	0:10:34	0:10:48	0:10:32 - 0:11:45
drosophila_38	99.992	99.975	99.992 - 99.993	99.876	99.871	99.823 - 99.920	0:12:04	0:12:01	0:11:46 - 0:12:09
drosophila_59	99.989	99.940	99.985 - 100.000	99.961	99.915	99.942 - 99.991	0:36:35	0:44:24	0:27:39 - 0:45:42
drosophila_60	99.964	99.152	99.963 - 99.998	99.896	99.076	99.878 - 99.897	0:22:37	0:27:02	0:13:14 - 0:23:23
drosophila_72	99.996	99.997	99.995 - 99.999	99.984	99.982	99.980 - 99.984	0:33:35	0:27:40	0:18:45 - 0:33:43
drosophila_80	99.975	99.972	99.967 - 99.978	99.878	99.879	99.877 - 99.879	1:54:15	1:51:19	1:48:36 - 1:54:33
drosophila_100_1	99.895	99.911	99.867 - 99.940	99.752	99.781	99.731 - 99.764	1:08:56	1:01:28	0:48:35 - 1:10:18
drosophila_100_2	99.998	99.558	97.800 - 100.000	99.971	99.530	97.775 - 99.976	0:32:32	0:41:57	0:31:19 - 1:19:35
drosophila_100_3	99.740	99.376	98.688 - 99.787	99.618	99.236	98.492 - 99.686	2:17:32	2:16:44	1:52:39 - 2:29:40
mice_2_1	99.850	99.853	99.837 - 99.883	99.100	99.145	99.012 - 99.255	0:03:15	0:03:13	0:03:10 - 0:03:17
mice_2_2	99.886	99.853	99.818 - 99.922	98.680	98.536	98.406 - 98.995	0:03:11	0:03:14	0:03:09 - 0:03:13
mice_2_3	100.000	100.000	100.000 - 100.000	98.932	98.951	98.805 - 99.071	0:03:03	0:03:02	0:02:47 - 0:03:33
mice_2_4	100.000	100.000	100.000 - 100.000	99.945	99.953	99.938 - 99.971	0:02:53	0:02:51	0:02:34 - 0:03:03
mice_4_1	99.893	99.850	99.640 - 99.940	99.716	99.685	99.488 - 99.770	0:03:19	0:03:11	0:03:03 - 0:03:26
mice_4_2	99.495	99.538	99.333 - 99.832	99.241	99.318	99.170 - 99.700	0:04:09	0:04:16	0:03:39 - 0:04:58
zebrafish_5	99.998	99.997	99.997 - 99.998	99.984	99.980	99.968 - 99.985	0:01:23	0:01:23	0:01:21 - 0:01:30
zebrafish_7	99.965	99.965	99.963 - 99.966	99.916	99.914	99.903 - 99.921	0:02:02	0:02:05	0:01:56 - 0:02:13
zebrafish_10_1	100.000	100.000	100.000 - 100.000	100.000	100.000	100.000 - 100.000	0:08:37	0:08:50	0:08:32 - 0:09:14
zebrafish_10_2	100.000	100.000	100.000 - 100.000	99.992	99.991	99.992 - 99.994	0:09:47	0:09:48	0:09:40 - 0:10:03
zebrafish_10_3	100.000	99.999	99.998 - 100.000	99.997	99.996	99.991 - 99.999	0:11:32	0:11:34	0:11:25 - 0:12:02
zebrafish_10_4	99.998	99.993	99.993 - 99.999	99.990	99.990	99.987 - 99.996	0:02:08	0:02:06	0:01:56 - 0:02:09
zebrafish_20	99.999	99.995	99.997 - 99.999	99.914	99.913	99.880 - 99.933	0:03:47	0:03:42	0:03:38 - 0:03:50
zebrafish_60_1	99.963	99.978	99.963 - 100.000	99.960	99.974	99.960 - 99.997	0:20:40	0:21:16	0:20:19 - 0:22:24
zebrafish_60_2	99.994	99.973	99.978 - 99.999	99.975	99.957	99.956 - 99.992	0:34:05	0:31:39	0:31:36 - 0:34:07
zebrafish_60_3	99.999	99.984	99.998 - 99.999	99.965	99.950	99.963 - 99.965	0:32:51	0:33:24	0:32:45 - 0:36:21
zebrafish_80_1	99.998	99.998	99.997 - 99.999	99.987	99.988	99.987 - 99.989	0:30:15	0:34:42	0:29:02 - 0:42:19
zebrafish_80_2	99.978	99.981	99.955 - 99.996	99.974	99.978	99.952 - 99.994	0:29:50	0:29:29	0:27:30 - 0:31:02
zebrafish_80_3	99.998	99.967	99.994 - 100.000	99.983	99.956	99.983 - 99.990	0:30:39	0:34:28	0:28:39 - 0:53:04
zebrafish_100_1	99.986	99.982	99.966 - 99.997	99.960	99.951	99.938 - 99.960	1:31:06	1:31:54	1:30:21 - 1:38:11
zebrafish_100_2	99.910	99.930	99.832 - 99.997	99.905	99.924	99.825 - 99.991	0:37:33	0:41:29	0:35:35 - 0:59:00
zebrafish_100_3	99.975	99.956	99.842 - 99.991	99.969	99.947	99.831 - 99.982	0:37:47	0:46:11	0:36:05 - 1:05:10

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	99.940	99.917	99.831 - 99.973	99.874	99.844	99.682 - 99.962	0:26:55	0:27:18	0:21:45 - 0:32:45
drosophila_8	94.665	94.253	88.727 - 99.994	94.680	94.128	88.392 - 99.988	0:31:08	0:31:25	0:28:55 - 0:32:19
drosophila_10	97.934	98.330	97.899 - 97.940	97.936	98.331	97.901 - 97.942	0:37:17	0:37:05	0:34:00 - 0:37:23
drosophila_38	99.915	99.839	99.666 - 99.916	99.807	99.680	99.371 - 99.824	1:34:29	1:20:44	1:25:16 - 1:42:35
drosophila_59	99.862	99.357	97.554 - 99.990	99.823	99.342	97.556 - 99.982	0:52:17	0:59:53	0:22:28 - 1:27:04
drosophila_60	99.998	99.987	99.943 - 99.998	99.998	99.987	99.943 - 99.998	1:05:04	1:10:01	0:58:33 - 1:14:41
drosophila_72	97.793	96.412	91.004 - 99.971	97.789	96.405	90.994 - 99.968	1:02:38	1:16:24	0:49:20 - 1:49:13
drosophila_80	97.643	97.742	95.802 - 99.230	97.618	97.702	95.735 - 99.183	1:36:49	1:38:19	1:20:36 - 1:49:03
drosophila_100_1	99.955	97.355	94.311 - 99.959	99.954	97.336	94.256 - 99.958	2:08:46	1:48:20	0:55:32 - 2:21:53
drosophila_100_2	74.957	73.651	60.365 - 85.480	74.930	73.642	60.371 - 85.469	0:49:40	0:50:59	0:44:09 - 0:53:05
drosophila_100_3	94.200	93.884	92.066 - 96.935	94.123	93.796	92.021 - 96.836	1:02:03	1:13:05	1:01:52 - 1:19:57
mice_2_1	99.683	99.419	99.611 - 99.687	98.515	98.122	97.791 - 98.628	0:05:46	0:05:40	0:05:15 - 0:05:52
mice_2_2	99.118	98.332	97.894 - 99.735	96.376	94.921	92.830 - 96.941	0:04:06	0:04:05	0:04:01 - 0:04:22
mice_2_3	99.775	99.301	98.821 - 99.801	97.601	96.930	95.688 - 97.788	0:04:03	0:04:04	0:03:55 - 0:04:08
mice_2_4	95.925	95.711	95.793 - 96.043	95.075	94.813	94.757 - 95.403	0:02:45	0:03:25	0:02:03 - 0:04:52
mice_4_1	99.964	99.959	99.940 - 99.965	99.577	99.574	99.559 - 99.581	0:18:11	0:18:10	0:17:39 - 0:18:31
mice_4_2	93.100	92.228	88.715 - 93.329	92.680	91.778	88.091 - 93.024	0:20:34	0:18:34	0:07:47 - 0:23:14
zebrafish_5	100.000	100.000	100.000 - 100.000	100.000	100.000	100.000 - 100.000	0:09:57	0:09:48	0:09:17 - 0:10:39
zebrafish_7	99.982	99.987	99.981 - 99.996	99.981	99.986	99.979 - 99.996	0:08:37	0:09:02	0:06:41 - 0:11:34
zebrafish_10_1	99.864	99.778	99.858 - 99.912	99.852	99.772	99.848 - 99.910	0:13:57	0:13:45	0:12:26 - 0:14:43
zebrafish_10_2	99.972	99.926	99.774 - 99.985	99.968	99.923	99.773 - 99.984	0:22:07	0:21:59	0:21:03 - 0:23:23
zebrafish_10_3	99.869	99.643	98.680 - 99.998	99.861	99.636	98.679 - 99.997	0:37:51	0:31:47	0:19:06 - 0:41:44
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	99.842	99.717 - 99.967	0:43:02	0:48:25	0:36:57 - 1:03:46
zebrafish_60_1	99.887	99.919	99.885 - 99.964	99.881	99.914	99.878 - 99.963	1:43:58	1:43:10	1:37:48 - 1:46:02
zebrafish_60_2	99.541	98.727	95.295 - 99.674	99.520	98.710	95.268 - 99.657	1:30:15	1:20:03	0:42:10 - 1:38:18
zebrafish_60_3	99.229	99.356	99.091 - 99.795	99.223	99.352	99.089 - 99.790	1:39:17	1:31:01	0:59:17 - 1:45:48
zebrafish_80_1	99.655	99.697	99.628 - 99.657	99.644	99.686	99.619 - 99.645	1:23:48	1:26:30	1:20:59 - 1:30:03
zebrafish_80_2	99.689	99.688	99.683 - 99.746	99.688	99.685	99.681 - 99.744	1:38:38	1:36:18	1:26:50 - 1:42:06
zebrafish_80_3	99.789	99.653	99.719 - 99.977	99.781	99.643	99.712 - 99.971	1:36:15	1:34:16	1:33:04 - 1:39:45
zebrafish_100_1	99.565	99.059	98.559 - 99.731	99.547	99.028	98.540 - 99.703	1:36:42	1:19:48	0:59:43 - 1:41:22
zebrafish_100_2	97.987	98.316	97.750 - 98.179	97.975	98.306	97.734 - 98.169	1:06:55	1:10:20	1:04:29 - 1:16:39
zebrafish_100_3	99.293	98.700	99.178 - 99.836	99.285	98.689	99.170 - 99.832	1:15:39	1:13:43	1:11:46 - 1:42:34

Supplementary Table 4. Performance of TRex in the benchmark.

Appendix 1

Preliminary concepts

Image-based tracking relies on identifying individuals through their visual features. The process begins by distinguishing the pixels corresponding to animals from those of the background. Let b represent a blob that is distinct from the background. For each blob b segmented from a video, an identification image I_b is generated by first taking the minimal bounding box image around b and then converting all pixels in I_b that do not belong to b to black. The blob within I_b is then rotated so that its first principal component is aligned at a $\frac{\pi}{4}$ angle to the x-axis and, finally, the image is cropped to a specified square size suitable for batch processing.

Each image I_b is classified as either an individual or a crossing of individuals. For more details on the background subtraction and individual-crossing classification process, please refer to Appendix D1-2 of the Supplementary Information of ***Romero-Ferrero et al.*** (***2019***).

A Fragment F is defined as a sequence of blobs that maintain a one-to-one spatial overlap, meaning they share pixels in each pair of consecutive frames over time. If two blobs merge into a single blob in the subsequent frame, or if a single blob splits into two in the next frame, each of these three blobs will terminate or initiate a new Fragment. Fragments are classified as either individual or crossing Fragments based on the classification of the blobs they contain. Blobs of different classifications are not permitted within the same Fragment. Since crossings are solved as a post-processing step after identification, from now on we will not take into consideration crossing Fragments, and we will refer to individual Fragments as Fragments.

A pair of Fragments is said to coexist if they both contain blobs from the same frames in the video. Moreover, being N the number of individuals in a video, a Global Fragment is defined as a collection of N Fragments all sharing a common frame.

By construction, we can assume that all blobs in a Fragment correspond to the same identity, this is the Fragment's identity. From this, coexisting Fragments will have different identities and Global Fragments will have all identities, one per Fragment.

From now on, we will denote F_i as the fragment with some arbitrary unique identifier i and I_{ik} will correspond to the identification image with the unique arbitrary identifier k in the fragment i.

General overview of Identification Protocols in the original idtracker.ai

In this section we will give a brief and high level overview on the algorithm idtracker.ai uses to assign identities to the different fragments. Please refer to ***Romero-Ferrero et al.*** (***2019***) for a more complete description of the algorithm.

Cascade of Training and Identification Protocols

The identification process begins with three sequential protocols that incrementally refine the identification network's ability to label individuals. The protocols leverage segments of the video where individuals appear distinctly, called global fragments, to construct a labeled dataset for the training of the network.

Protocol 1: Basic Accumulation of Global Fragments In Protocol 1, the algorithm searches for global fragments. The initial set of labeled images from these fragments forms the base dataset to train the identification network. This trained network is then used to label additional global fragments throughout the video. If Protocol is not able to accumulate at least 99.95% of all images in the global fragments, the algorithm proceeds to Protocol 2.

Protocol 2: Iterative Expansion with High-Quality Fragments Protocol 2 builds on the initial training by iteratively alternating between accumulating new global fragments and using them to further train the identification network. With each iteration, the network labels more fragments, adding only those that pass strict quality checks (explained in the section below). This process continues until either 99.95% of the images in the global fragments are labeled with high certainty, or no more high-quality fragments are available.

Protocol 3: Pretraining and Fine-Tuning for Complex Scenarios Protocol 2 might fail for videos with high visual complexity (accumulating less than 90% of the images). In those cases, idtracker.ai proceeds to Protocol 3. Protocol 3 pretrains the convolutional layers of the identification network on a large sample of global fragments, using the same convolutional layers for each global fragment while changing only the last classification layer. Although this protocol is effective in tracking videos that cannot be tracked with Protocol 2, it is very slow and may take days for some videos.

Labeling and Accumulating Images in Global Fragments

The process of labeling and accumulating images from global fragments involves the following steps:

1. **Selection of Global Fragments:** The algorithm identifies global fragments where all animals are visually distinct, ensuring unambiguous initial identity assignments.
2. **Labeling with the Trained Network:** The identification network, trained on an initial set of global fragments, predicts identities across additional fragments belong to the other global fragments. Each fragment is assigned an identity based on the network's classification probabilities of its corresponding images, denoted $P1(F, i)$.
3. **Quality Checks:** Labeled fragments are subjected to a series of quality checks to ensure the reliability of their identity assignments. For each global fragment these checks include:
   - **Certainty:** Each fragment F must have a high certainty score, defined by the distinction between the highest and second-highest identity probabilities:

     $$\text{cert}(F) = \frac{\text{median}(S_a) \cdot P1(F, a) - \text{median}(S_b) \cdot P1(F, b)}{P1(F, a) + P1(F, b)}$$

     where $P1(F, i)$ represents the probability of fragment F being assigned identity i. Here, a and b represent the identity predictions with the highest and second highest $P1$ values for fragment for F, with S_a and S_b being the vectors of softmax values of all the images in the fragment F assigned to the identities a and b respectively.
   - **Consistency:** The identity assignment for each fragment must remain consistent across frames, preventing arbitrary changes in identity due to minor variations in appearance. This is reflected on the value of $P1$.
   - **Uniqueness:** Within a single global fragment, each assigned identity must be unique, ensuring that no two animals share the same identity label within that fragment.
4. **Accumulation into the Training Set:** Fragments that pass the quality checks are added to the training dataset, allowing the network to improve its accuracy iteratively. This accumulation process continues, increasing the network's generalization ability across the video.

Residual Identification

After the cascade protocols, residual identification is applied to label any fragments that remain unlabeled or have low-certainty assignments. This step uses a probabilistic approach that accounts for temporal coexistence constraints, refining identity assignments. For each unlabeled fragment F, an adjusted probability $P2(F, i)$ is computed for assigning identity i, considering neighboring fragments $\gamma(F)$ that overlap in time:

$$P2(F, i) = \frac{P1(F, i) \prod_{G \in \gamma(F)} (1 - P1(G, i))}{\sum_j P1(F, j) \prod_{G \in \gamma(F)} (1 - P1(G, j))}$$

where $P1(F, i)$ represents the initial probability of F being identity i.

Afterwards a new measure of identification certainty is defined as

$$\overline{\text{cert}}(F) = \frac{P2(F, a)}{P2(F, b}$$

in which a and b again represent the identity predictions with the highest and second highest $P1$ values for fragment for F. Fragments then are assigned identities in descending order of certainty, with the highest-confidence fragments labeled first.

In this work, the primary advancement was the replacement of protocols in idtracker.ai with an identification method based on deep metric learning. Additionally, several smaller but significant technical improvements were implemented, enhancing feature set, tracking time, and memory usage efficiency.

Appendix 2

Contrastive protocol

Contrastive learning is a type of self-supervised learning that aims to learn useful data representations by contrasting positive and negative pairs of examples. The fundamental idea is to bring similar (positive) pairs closer in the representation space while pushing dissimilar (negative) pairs farther apart. This approach leverages the inherent structure of the data, allowing the model to learn without labeled examples.

The representation space or embedding in contrastive learning is a high-dimensional environment where data points are mapped to vectors, capturing essential features and patterns of the original data. This space can be conceptualized as a vast, multidimensional environment in which each data point is represented as a vector. The primary objective is to position similar data points in close proximity while ensuring that dissimilar data points are situated at a considerable distance from one another. Positive pairs are typically created by applying different transformations or augmentations to the same data point, such as cropping, rotating, or color jittering an image, preserving the inherent semantics of the original data point. These augmentations ensure that the model learns robust features invariant to such transformations. Conversely, negative pairs are composed of different data points expected to be dissimilar, such as two distinct images.

As the model undergoes training, the representation space becomes increasingly structured, with similar types of data points forming coherent clusters. These clusters encapsulate the inherent similarities within the data, even if the specific instances differ, such as different breeds of cats or different poses. By maximizing the agreement between positive pairs and minimizing the agreement between negative pairs, the model learns to distinguish subtle differences and similarities within the data. The contrastive loss minimizes the distance between positive pairs and maximizes the distance between negative pairs in the representation space. This contrastive objective ensures the learned representations capture essential features and discriminative patterns, facilitating downstream tasks such as classification, clustering, and retrieval, even without labeled data. Thus, the representation space serves as a learned map where the positions of data points reflect their semantic relationships, enabling the model to capture and utilize the underlying structure of the data for various tasks.

We apply the principles of contrastive learning to create an embedding of all the images in a video that reflects the fragmented structure of the video. Specifically, points in the embedding corresponding to images from coexisting fragments (different identities) are positioned further apart than points corresponding to images from the same fragment (same identity) (***Figure 1a–c***).

1. **Segmentation and Fragmentation**: The video is segmented and the blobs grouped into fragments based on temporal or content-based criteria.
2. **Training ResNet18**: ResNet18 is trained using positive pairs (images from the same fragment) and negative pairs (images from coexisting fragments). The network learns a representation space where the distance between positive pairs is minimized, while the distance between negative pairs is maximized.
3. **Clustering in the Representational Space**: All images are passed through the network. K-means clustering is then applied to the embedded images, assigning them to different cluster labels.
4. **Cluster based labeling of Single Image**: Each cluster is labeled as a distinct animal identity. Images are classified based on their assigned clusters, and a probability dis-

tribution for each identity prediction is computed based on the Euclidean distance to the center of each cluster. If global fragments are present, proceed to next step; otherwise, proceed to Step 7.

5. **Fragment Identification with Global Fragments**: A thorough identification process is conducted to classify all images belonging to global fragments, correcting any errors from the initial classification. If 99.9% > of all the images in global fragments are successfully accumulated (pass the quality checks, see section 1), go to Step 7; otherwise, go to next step.
6. **Run Accumulation Protocol if Step 5 Fails**: Run protocol 2 from idtracker.ai v5 but using correctly identified images as the ground truth, as a sort of synthetic first Global Fragment.
7. **Residual Identification**: A thorough identification process is conducted to classify all images in the video, correcting any errors from the initial classification step.

Network architecture

Deep metric learning often requires larger networks for classification tasks compared to standard supervised learning. To identify the most suitable architecture, we evaluated several state-of-the-art image classification networks, including the model used in the original idtracker.ai.

There were specific constraints in selecting the optimal architecture. The image size is automatically set during each tracking session to fit the average blob size, but it is typically small, ranging from 20×20 to 100×100 pixels. This limited some architectures, such as AlexNet, which requires a fixed input size of 227×227, and DenseNet, which has a minimum input size of 29×29. Additionally, the large training batches commonly associated with deep metric learning necessitate a compact model that can be trained on a consumer-grade GPU. This constraint excluded other architectures, including EfficientNet and the larger ResNet models (ResNet101 and ResNet152).

As shown in ***Figure 1—figure Supplement 1***, ResNet18 offered the best balance between training speed and tracking accuracy.

Embedding dimension

Another critical hyperparameter is the embedding dimension. Here, too, there is a trade-off between achieving a robust representation of subtle differences between animals—differences that may be minimal and even challenging to detect visually—and maintaining a compact network size and efficient training speed. This parameter was empirically determined to be 8 (***Figure 1—figure Supplement 2***).

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. Mathematically for our case, the contrastive loss L for a pair of images (I_{ik}, I_{jl}) is defined as:

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

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 are crucial in our problem, where we aim to embed individuals of the same identity in similar regions of the representational space. However, we face the restriction of not being able to 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.

In the contrastive protocol, we used $D_{\text{pos}} = 1$ and $D_{\text{neg}} = 10$. These values were determined empirically and provide effective embeddings and were robust for tracking multiple videos across various species and different numbers of animals (***Figure 1—figure Supplement 3***).

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. 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 global fragments are present, follow the procedure mentioned in the subsection 1. If on the contrary there are no global fragments we move straight to residual identification as explained in section 1

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

$$P\left(I_{ik} \text{ belongs to identity } j\right) = \frac{d_j(I_{ik})^7}{\sum_j d_j(I_{ik})^7} \tag{2}$$

Equation (2) 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 global fragments exist, we use them for K-means initialization: we use the embeddings from the first global fragment as initial cluster centers, choosing the one where the minimum fragment is the largest. This approach provides a strong initialization for the K-means algorithm, aligning it with the different identities and mitigating issues related to random initialization. It also allows us to better compare clusters as training progresses.

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

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

and the mean nearest-cluster distance

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

The SS is given by

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

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 1d*** and ***Figure 1e***). The number of images used for the computation of the SS is 1000 times the number of animals.

Pairs selection

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: first, we randomly select a pair of coexisting fragments, and then we 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.

Empirical testing has revealed that large and balanced batches, with an equal number of positive and negative pairs, are ideal for our setting of contrastive learning. More concretely,

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 didn't compromise training speed/accuracy (***Figure 1—figure Supplement 4***). 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.

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:

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

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

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

By examining the evolution of the clusters during training (***Figure 1c***) it becomes clear that the learning process is not uniform; some identities become separated sooner than others. ***Figure 1c*** top row second and third columns give us a nice illustration of this phenomenon. The images embedded in the red 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 green rectangle 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

$$l_s(i,j) \longleftarrow \begin{cases} (l_s(i,j)+1)(1-0.02), & \text{if } \mathcal{L}\left(I_{il}, I_{ik}\right) > 0 \\ l_s(i,j)(1-0.02), & \text{otherwise} \end{cases} \tag{3}$$

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 probably distribution over all pairs of fragments by

$$P_{l_s}(F_i, F_j) = \begin{cases} \frac{l_s(i,j)}{\sum_{i \neq j} l_s(i,j)}, & \text{if } i \neq j \\ \frac{l_s(i,i)}{\sum_i l_s(i,i)}, & \text{otherwise} \end{cases} \tag{4}$$

The final probability of sampling pairs of fragments is given by

$$P(F_i, F_j) = \alpha P_s(F_i, F_j) + \left(1-\alpha\right) P_{l_s}(F_i, F_j) \tag{5}$$

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 saw that a value of α around $\frac{1}{2}$ produced the best decrease the time required to train the network across a large collection of videos (***Figure 1—figure Supplement 5***). 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.

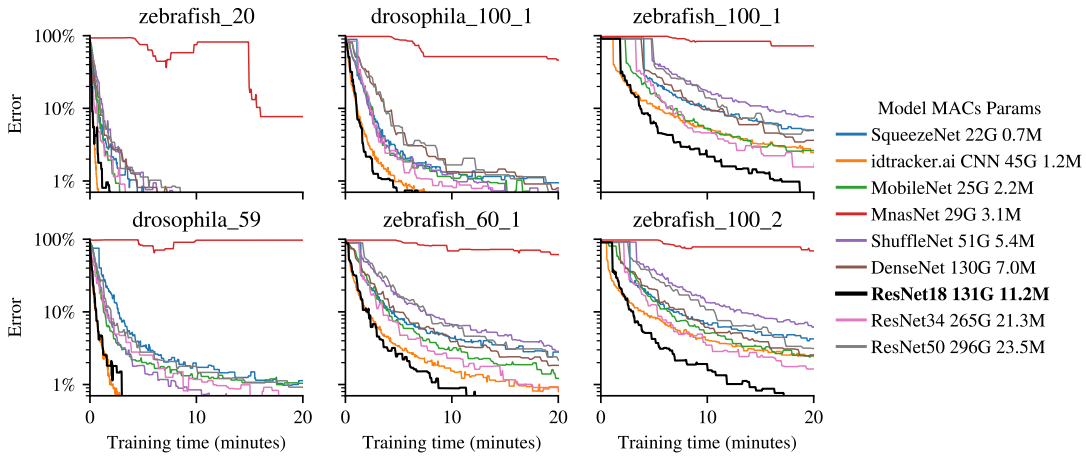

Figure 1—figure supplement 1. Models comparison. Error in image identification as a function of training time for different deep learning models in 6 test videos. For each network we report the multiply-accumulate operations (MAC) in giga operations (G) and the number of parameters in the units of million 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.

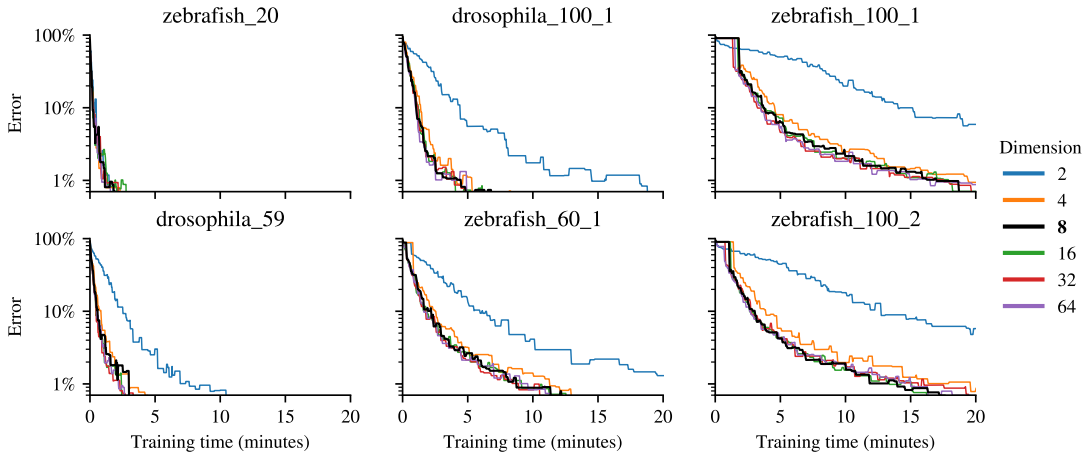

Figure 1—figure supplement 2. Embedding dimensions comparison. Error in image identification as a function of training time for different embedding dimensions in 6 test videos. Every 100 training batches, we perform k-means clustering on a randomly selected set of 20,000 images, assigning identities 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.

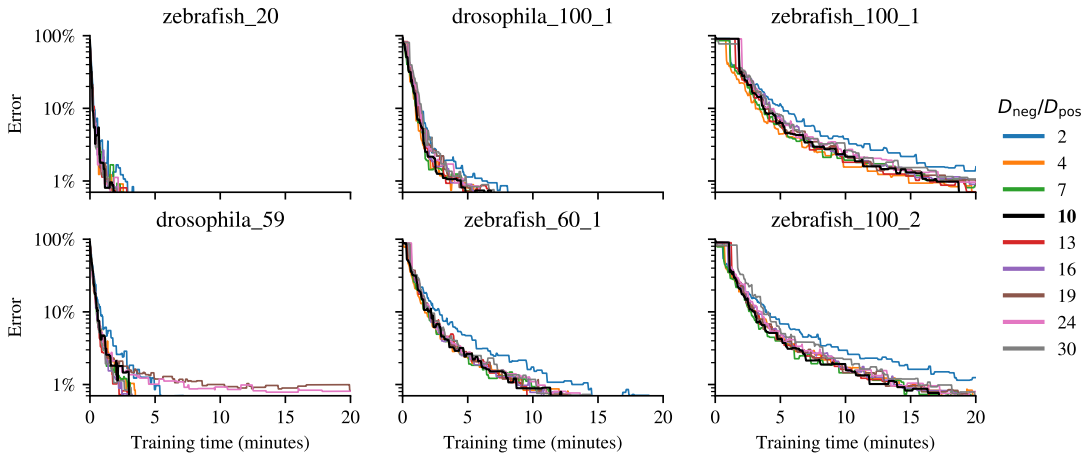

Figure 1—figure supplement 3. D_{neg} over D_{pos} comparison. Error in image identification as a function of training time for different ratios of D_{neg}/D_{pos} in 6 test videos. Every 100 training batches, we perform k-means clustering on a randomly selected set of 20,000 images, assigning identities 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.

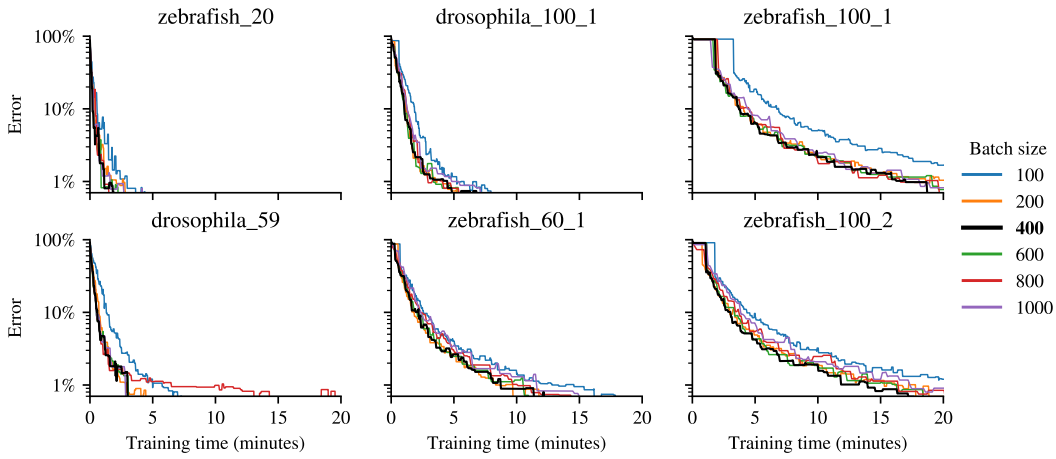

Figure 1—figure supplement 4. Batch size comparison. Error in image identification as a function of training time for different batch sizes of pairs of images in 6 test videos. Every 100 training batches, we perform k-means clustering on a randomly selected set of 20,000 images, assigning identities 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.

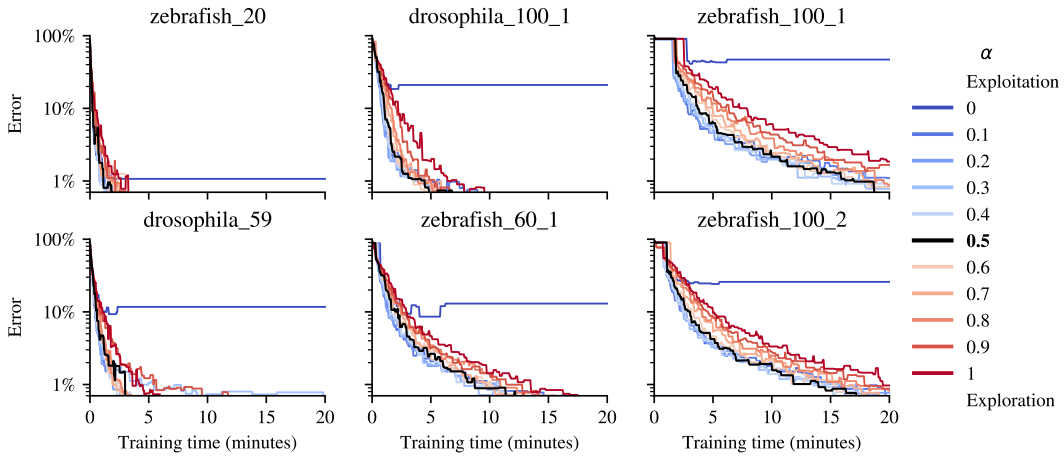

Figure 1—figure supplement 5. Exploration and exploitation comparison. Error in image identification as a function of training time for different exploration/exploitation weights α in 6 test videos. Every 100 training batches, we perform k-means clustering on a randomly selected set of 20,000 images, 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.

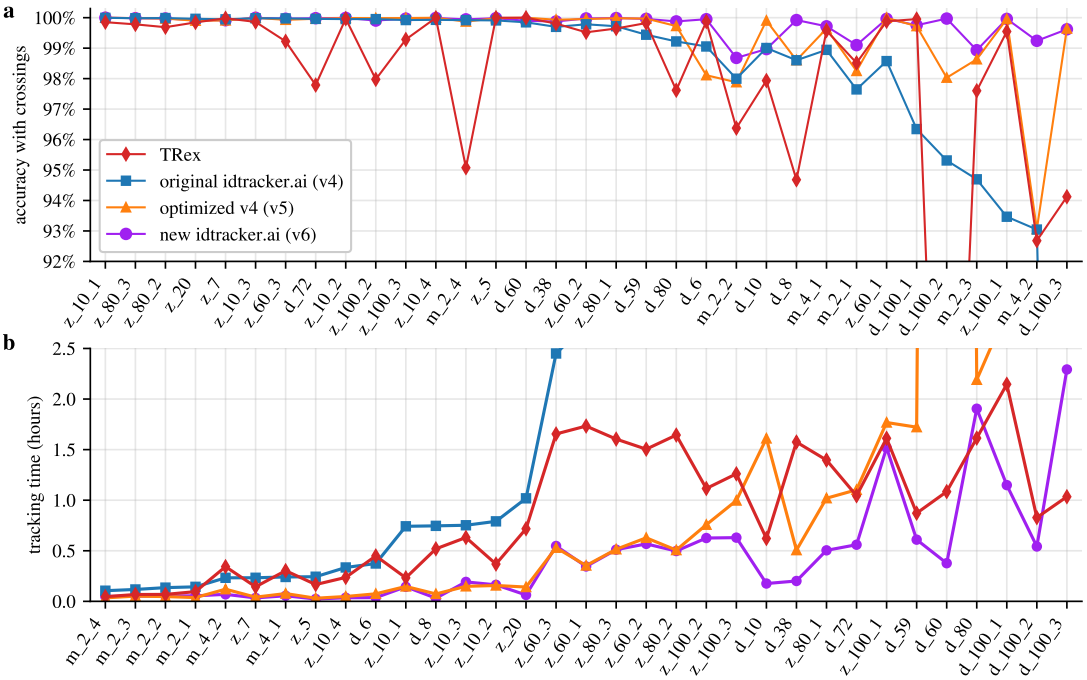

Figure 2—figure supplement 1. Performance for the benchmark with full trajectories with animal crossings. **a**. Median accuracy was computed using all images of animals in the videos including animal crossings. **b**. Median tracking times. **Supplementary Table 1**, **Supplementary Table 2**, **Supplementary 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.

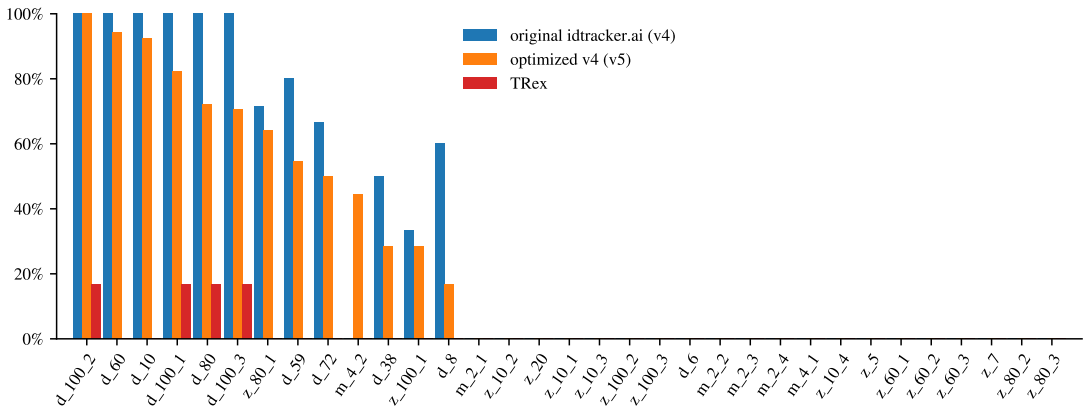

Figure 2—figure supplement 2. Protocol 2 failure rate. Probability for the different tracking systems of not tracking the video with Protocol 2 in idtracker.ai (v4 and v5) and in TRex the probability that it fails without generating trajectories.

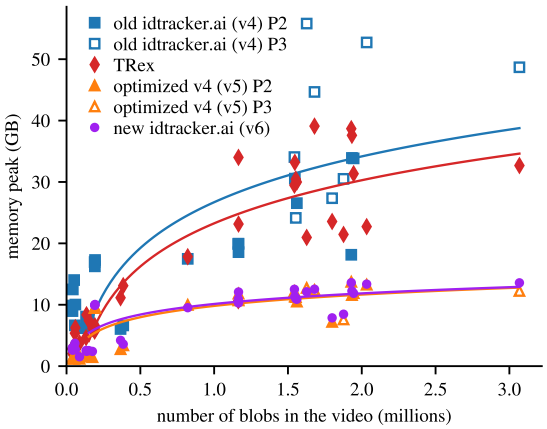

Figure 2—figure supplement 3. Memory usage across the different softwares. The solid line is a logarithmic fit to the memory peak as a function of the number of blobs in a video. Disclaimer: Both software programs include automatic optimizations that adjust based on machine resources, so results may vary on systems with less available memory. These results were measured on computers with the specifications in **Methods**

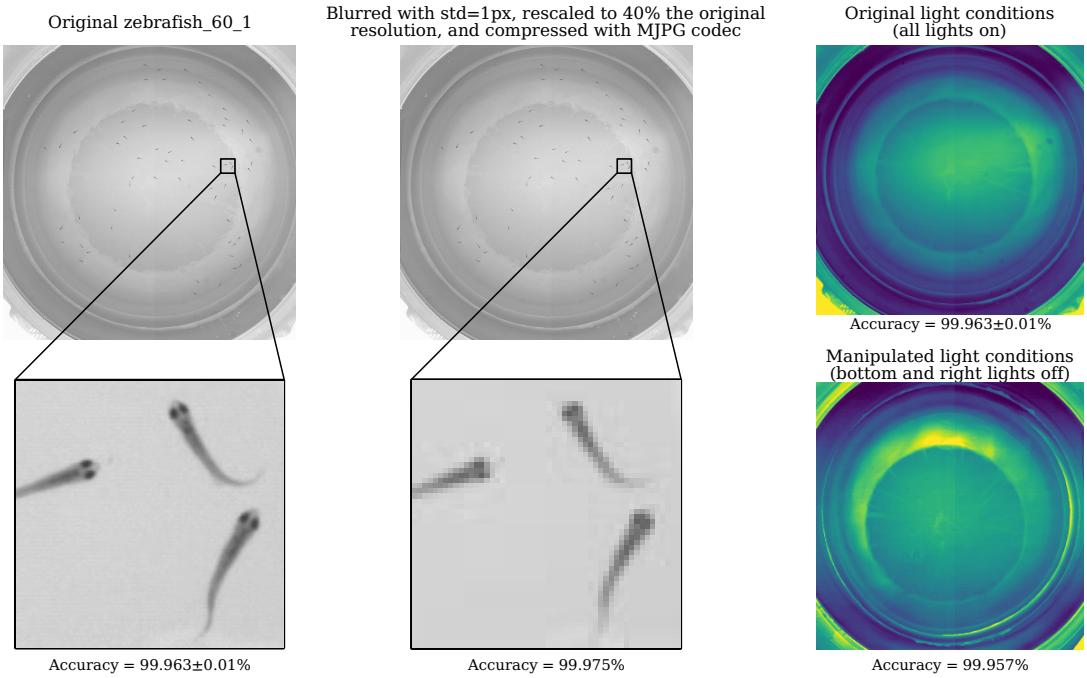

Figure 2—figure supplement 4. Robustness to blurring and light conditions. First column: Unmodified video zebrafish_60_1. **Second column:** zebrafish_60_1 with a gaussian blurring of sigma=1 pixel plus a resolution reduction to 40% of the original plus MJPG video compression. **Third column:** Videos 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.