nature methods

BRIEF COMMUNICATION

https://doi.org/10.1038/s41592-018-0295-5

# idtracker.ai: tracking all individuals in small or large collectives of unmarked animals

Francisco Romero-Ferrero[1,2], Mattia G. Bergomi[1,2], Robert C. Hinz[1], Francisco J. H. Heras[1] and Gonzalo G. de Polavieja[1*]

**Understanding of animal collectives is limited by the ability to track each individual. We describe an algorithm and software that extract all trajectories from video, with high identification accuracy for collectives of up to 100 individuals. idtracker.ai uses two convolutional networks: one that detects when animals touch or cross and another for animal identification. The tool is trained with a protocol that adapts to video conditions and tracking difficulty.**

Researchers attempting to determine animal trajectories from video recordings face the problem of maintaining correct animal identifications after individuals touch, cross or are occluded by environmental features. To bypass this problem, we previously implemented in idTracker the idea of tracking the identification of each individual by using a set of reference images obtained from the video[1]. idTracker and further developments in identification algorithms for unmarked animals[2–6] have been successful for small groups of 2–15 individuals and in situations with few crossings[5,7].

Here we present idtracker.ai, a species-agnostic system able to track all individuals in both small and large collectives (up to 100 individuals) with high identification accuracy—often greater than 99.9%. A graphical user interface walks users through tracking, exploration and validation (Fig. 1a). The system uses two different convolutional networks[8–10], as well as algorithms for species-agnostic preprocessing, extraction of training datasets from the video and post-processing (Fig. 1b).

The preprocessing module extracts 'blobs', areas of each video frame that correspond either to a single animal or to several animals that are touching or crossing. The blobs are then oriented according to their axes of maximum elongation (Fig. 1c).

The convolutional 'crossing detector' network determines whether each preprocessed blob corresponds to a single animal or to a crossing (Fig. 1d; network architecture in Supplementary Table 1). idtracker.ai trains this network using images that a set of heuristics labels with high confidence as single animals or crossings (Methods). Once trained, the network can classify all blobs as single animals or crossings.

The convolutional identification network is then used to identify each individual between two crossings (Fig. 1e; Supplementary Table 1 outlines the network architecture). We measured the identification capacity of this network using 184 single-animal videos, with 300 pixels per animal on average. We randomly selected 3,000 images per animal for training. Tests of the network with 300 new images showed >95% single-image accuracy for up to 150 animals (Fig. 1f; Supplementary Fig. 1 shows the experimental setup, and Supplementary Fig. 2 shows results obtained with the alternative architectures detailed in Supplementary Tables 2 and 3). By comparison, the accuracy of idTracker[1] degraded more quickly, to 83% for 30 individuals, and computationally the program is too demanding for larger groups.

With videos of collective behavior, however, we typically lacked direct access to 3,000 images per animal for training of the identification network. Hence, to obtain the training images, we developed a cascade of three protocols that were recruited sequentially depending on the difficulty of the video (Fig. 1b; Supplementary Figs. 3 and 4 show experimental setups with zebrafish and flies, and the Supplementary Notes provide details on the algorithms).

Protocol 1 first finds all intervals of the video where all the animals are detected as separate from one another. The protocol then extends the image fragments for each animal by adding the preceding and following image frames up to the previous and next crossing for each animal. We call these extended intervals global fragments; they can contain different numbers of images per animal. Subsequently, the system determines the shortest distance traveled by an individual animal within a global fragment and then chooses the global fragment in which this shortest distance is maximal across the dataset (Fig. 1g). The system then uses this global fragment to train the identification network. Once trained, the network assigns identities in all the remaining global fragments. Afterward, the system uses a set of heuristics to select those global fragments with a high-quality assignment of identities (Methods). If these high-quality global fragments (Fig. 1g) cover <99.95% of the images in the global fragments, then protocol 1 fails and protocol 2 starts.

Protocol 2 accumulates identified images in global fragments as training examples, without human intervention. It starts by retraining the network with the additional high-quality global fragments found in protocol 1. This new network then assigns the remaining global fragments, from which the system selects the high-quality ones. This procedure iterates, always converting high-quality test examples into training examples, until no more high-quality global fragments remain or until 99.95% of the images from global fragments are accumulated. Upon completion, protocol 2 can be declared successful if it finished by accumulating 99.95% of images from global fragments or if >90% of the images in global fragments had been accumulated at the point when no more high-quality global fragments were available. In our example, protocol 2 was successful at the ninth step, when it had accumulated 99.95% of images from global fragments (Fig. 1g).

Post-processing starts with assignment of the remaining images using the final network (Fig. 1h). Then, identification accuracy is estimated using a Bayesian framework that aggregates evidence from multiple individual images in each global fragment

[1]Champalimaud Research, Champalimaud Center for the Unknown, Lisbon, Portugal. [2]These authors contributed equally: Francisco Romero-Ferrero, Mattia G. Bergomi. *e-mail: gonzalo.polavieja@neuro.fchampalimaud.org

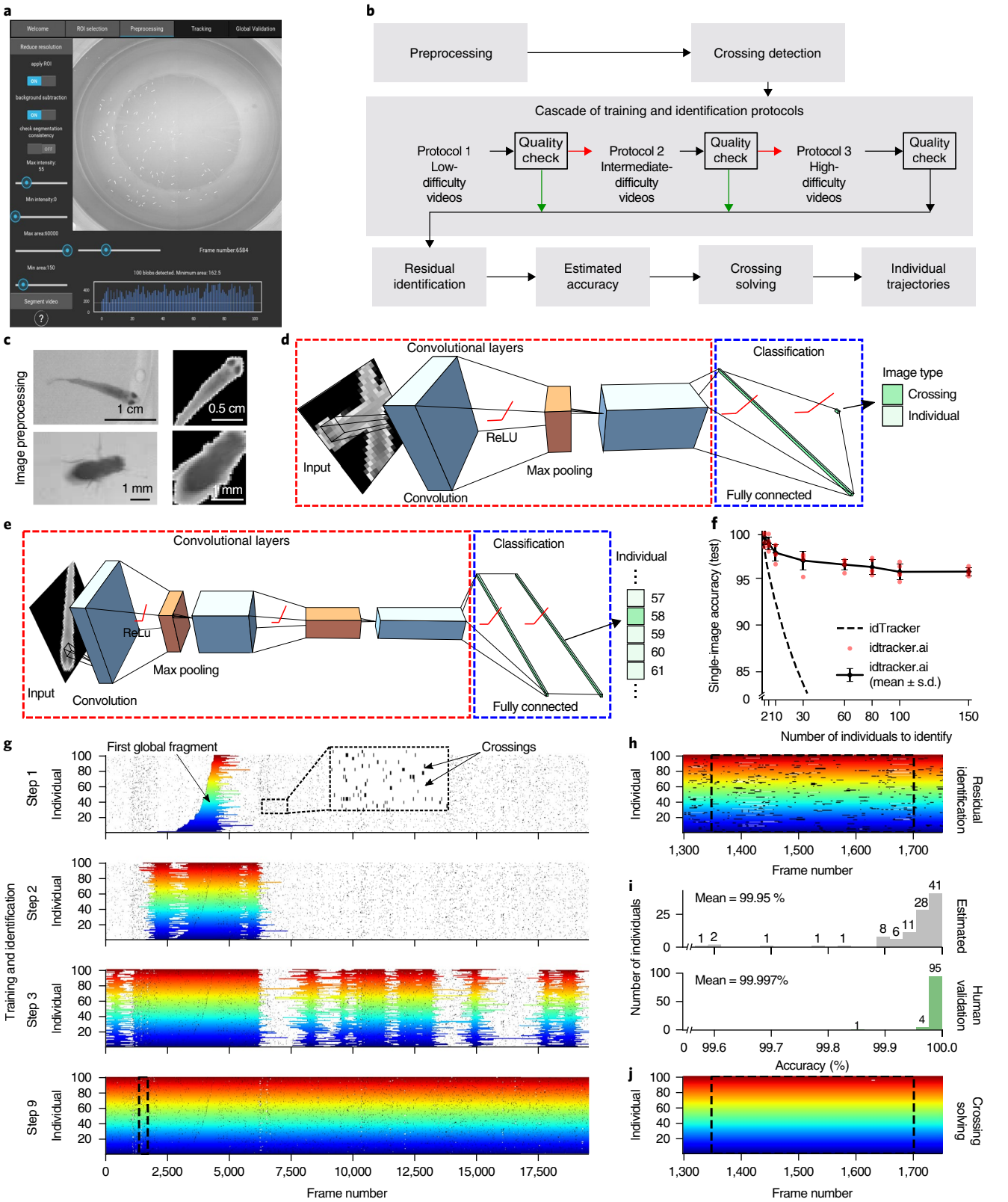



**Fig. 1 | Tracking by identification in idtracker.ai. a**, Graphical user interface. **b**, Diagram of the processing steps. **c**, Preprocessing extracts single-animal and multi-animal blobs. **d**, The crossing-detector network. ReLU, rectified linear units. **e**, The identification network. **f**, Single-image accuracy of idtracker.ai (solid line, mean ± s.d.; red dots, single trials; $n = 5$) and idTracker[1] (dashed line). **g**, Accumulation of training images in a video of 100 zebrafish at 31 days post-fertilization. Colors indicate each of the 100 individuals, and small vertical black segments indicate animal crossings. In step 1, the identification network trains on the starting global fragment and assigns the other global fragments; then a subgroup of high-quality global fragments is extracted (step 2). Protocol 2 increases the size of the training dataset by iterating training and quality checks, here ending at step 9. **h**, Identification of remaining small segments (lighter colors). **i**, Estimated (top) and human-validated (bottom) accuracies. **j**, Post-processing assignment of crossings (black) and very small nonassigned (white) segments from the data in **h**.

(Supplementary Fig. 5). In our example, the system estimated 99.95% accuracy at this step (Fig. 1i, top); human validation of 3,000 sequential video frames (680 crossings) gave 99.997% accuracy (Fig. 1i, bottom). Animal crossings are then resolved through iterative image erosion and interpolation[1] (Fig. 1j, Supplementary Notes). The human-validated accuracy was 99.988% for the final assignments, including images between and during crossings.

If protocol 2 fails, protocol 3 starts by pretraining only the convolutional part of the identification network, using most of the global fragments. In this first step, the system does not accumulate global fragments, and the classification layer is reinitialized after training with each global fragment. Then, it proceeds in the same way as protocol 2, but training only the classification layer and fixing the parameters of the convolutional layers at the values obtained in the first step. The different stages in the protocol cascade (protocols 1–3 and post-processing) add to the accuracy and computational time (Supplementary Table 4).

We tested idtracker.ai on small and intermediate-size groups of four species (Supplementary Table 5) and on large collectives of zebrafish and flies (Supplementary Table 6). In zebrafish, protocol 2 was always successful for large groups, giving accuracies of 99.96% ± 0.06% for 60 individuals and 99.99% ± 0.01% for 100 individuals (mean ± s.d.; $n = 3$ for both groups). With flies, the system applied protocol 3 for groups of more than 38 individuals and reached high accuracies (99.997%) for groups of up to 72 individuals. With groups of 80–100 flies, the system reached its limit, but still had >99.5% accuracy.

We have studied potential limitations of the system. One concern is how large the global fragments need to be. We typically find 300–1,000 global fragments in 10-min videos, with the extraction of each one requiring only one frame with no crossings. Our tests have shown that the pipeline is typically successful when it starts with a global fragment containing >30 images per animal, although it can work with fewer images (Supplementary Fig. 6). Videos of large collectives of up to 100 zebrafish were found to fulfill this condition of >30 images per animal by a large margin (Supplementary Fig. 6). Videos of flies also worked with this number of images, except in recordings of very low locomotor activity acquired in a low-humidity setup (Supplementary Fig. 6, Supplementary Table 7).

A second concern is how performance depends on image quality. We recommend working with around 300 pixels per animal at the segmentation step, but our tests indicated good performance with as few as 25 pixels per animal, which corresponds to 100 pixels per animal at the identification stage owing to a dilation of segmented animals (Supplementary Fig. 7 and Supplementary Table 8). Also, the system is robust to blurring (Supplementary Fig. 8 and Supplementary Table 9), inhomogeneous lighting (Supplementary Fig. 9) and image-compression algorithms (Supplementary Table 10). A reduction of image quality typically increases the computational time (Supplementary Tables 8 and 9). Shorter computational times can be achieved through transfer learning[11] (Supplementary Table 11, Methods).

Finally, we illustrate the use of idtracker.ai (Fig. 2). Using an attack score (Methods) applied to two adult male zebrafish staged so as to trigger fight behavior[12] (Fig. 2a), we found the expected pattern of frequent attacks from both fish followed by one dominating (Fig. 2b, top), but also reversals of dominance (Fig. 2b, middle) and dominance of one animal from the start (Fig. 2b, bottom; Supplementary Fig. 10). idtracker.ai also tracked a group of 14 ants, *Diacamma indicum*, despite shadows, light reflections and immobile animals (Fig. 2c). Active ants activate immobile ants (Fig. 2d) in direct proportion to their level of activity (Fig. 2e; Pearson's $R^2 = 0.75$, two-sided Wald test $P = 6 \times 10^{-5}$). We noted that 100 juvenile zebrafish formed mills (Fig. 2f). Different individuals visited the arena differently, and those who preferred the periphery moved faster (Fig. 2g,h; Pearson's $R^2 = 0.58$, two-sided Wald test $P = 3 \times 10^{-20}$; see Supplementary Fig. 11).

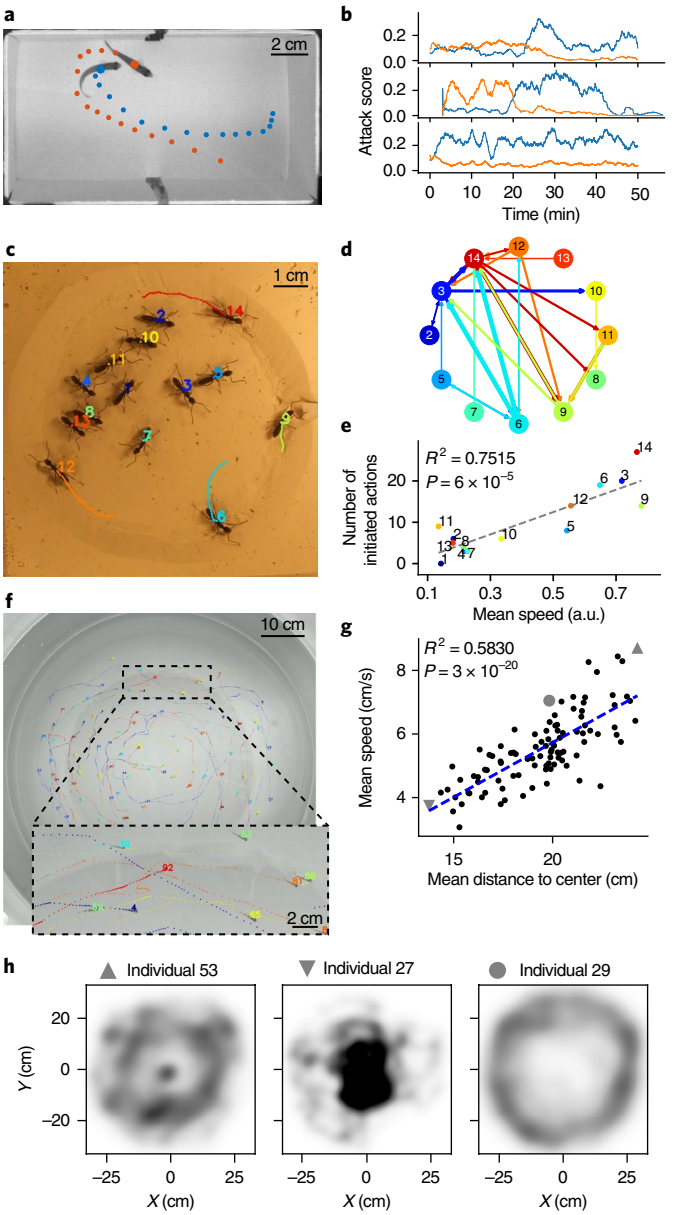



**Fig. 2 | Using idtracker.ai to study small and large animal groups. a**, Two adult male zebrafish staged so as to induce fighting behavior. Dotted lines represent small portions of their trajectories. **b**, Attack scores versus time for two individual male zebrafish. The three plots correspond to three different pairs of fish. **c**, Photo of 14 ants. Photo is a frame from a video courtesy of A. I. Bruce (Monash University, Melbourne, Australia) and N. Blüthgen (Technische Universität Darmstadt, Darmstadt, Germany) tracked with idtracker.ai. **d**, Network of ant interactions. An arrow connects two individuals if the locomotor activity of a source individual caused a response in a target individual. Line thickness represents the frequency of the interactions. **e**, Correlation of the number of locomotor initiations an animal produces with its mean speed. a.u., arbitrary units. **f**, Frame of 100 juvenile zebrafish, and a zoomed-in view of a smaller group of those fish. **g**, Correlation between mean speed and mean distance to the center of the arena. Each dot represents one of the 100 animals in the collective shown in **f**; gray symbols correspond to the individual animals references in **h**. **h**, Probability density of finding an individual in a certain position in the arena for three different fish.

## Online content

Any methods, additional references, Nature Research reporting summaries, source data, statements of data availability and associated accession codes are available at https://doi.org/10.1038/s41592-018-0295-5.

Received: 30 June 2018; Accepted: 26 November 2018;
Published online: 14 January 2019

## References

1. Pérez-Escudero, A., Vicente-Page, J., Hinz, R. C., Arganda, S. & de Polavieja, G. G. *Nat. Methods* **11**, 743–748 (2014).
2. Dolado, R., Gimeno, E., Beltran, F. S., Quera, V. & Pertusa, J. F. *Behav. Res. Methods* **47**, 1032–1043 (2015).
3. Rasch, M. J., Shi, A. & Ji, Z. *bioRxiv* Preprint at https://www.biorxiv.org/content/early/2016/08/24/071308 (2016).
4. Rodriguez, A., Zhang, H., Klaminder, J., Brodin, T. & Andersson, M. *Sci. Rep.* **7**, 14774 (2017).
5. Wang, S. H., Zhao, J. W. & Chen, Y. Q. *Multimed. Tools Appl.* **76**, 23679–23697 (2017).
6. Xu, Z. & Cheng, X. E. *Sci. Rep.* **7**, 42815 (2017).
7. Lecheval, V. et al. *Proc. Biol. Sci.* **285**, 1877 (2018).
8. LeCun, Y., Bengio, Y. & Hinton, G. *Nature* **521**, 436–444 (2015).
9. Abadi, M. et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems. *TensorFlow.org* http://download.tensorflow.org/paper/whitepaper2015.pdf (2015).
10. Rusk, N. *Nat. Methods* **13**, 35 (2016).
11. Pan, S. J. et al. *IEEE Trans. Knowl. Data Eng.* **22**, 1345–1359 (2010).
12. Laan, A., Iglesias-Julios, M. & de Polavieja, G. G. *R. Soc. Open Sci.* **5**, 180679 (2018).

## Acknowledgements

We thank A. Groneberg, A. Laan and A. Pérez-Escudero for discussions; J. Baúto, R. Ribeiro, P. Carriço, T. Cruz, J. Couceiro, L. Costa, A. Certal and I. Campos for assistance in software, arena design and animal husbandry; and A. Bruce (Monash University, Melbourne, Australia), N. Blüthgen (Technische Universität Darmstadt, Darmstadt, Germany), C. Ferreira, A. Laan and M. Iglesias-Julios (Champalimaud Foundation, Lisbon, Portugal) for videos of ants, flies and zebrafish fights. This study was supported by Congento LISBOA-01-0145-FEDER-022170, NVIDIA (M.G.B., F.H. and G.G.d.P.), PTDC/NEU-SCC/0948/2014 (G.G.d.P.) and Champalimaud Foundation (G.G.d.P.). F. R.-F. acknowledges an FCT PhD fellowship.

## Author contributions

F.R.-F., M.G.B. and G.G.d.P. devised the project and algorithms and analyzed data. F.R.-F. and M.G.B. wrote the code with help from F.H. M.G.B. managed the code architecture and GUI. F.R.-F. managed testing procedures. R.H. built setups and conducted experiments with help from F.R.-F. G.G.d.P. supervised the project. M.G.B. wrote the supplementary material with help from F.R.-F., R.H., F.H. and G.G.d.P., and G.G.d.P. wrote the main text with help from F.R.-F., M.G.B. and F.H.

## Competing interests

The authors declare no competing interests.

## Additional information

**Supplementary information** is available for this paper at https://doi.org/10.1038/s41592-018-0295-5.

**Reprints and permissions information** is available at www.nature.com/reprints.

**Correspondence and requests for materials** should be addressed to G.G.d.P.

**Publisher's note:** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

## Methods

**Ethics.** Animal handling and experimental procedures were approved by the Champalimaud Foundation Ethics Committee (CF internal reference 2015/007) and the Portuguese Direcção Geral Veterinária (DGAV reference 0421/000/0002016) and were performed according to European Directive 2010/63/EU[13].

**Tested computer specifications.** We tracked all the videos with desktop computers running GNU/Linux Mint 18.1 64-bit (Intel Core i7-6800K or i7-7700K, 32 or 128 GB RAM, Titan X or GTX 1080 Ti GPUs, and 1 TB SSD disk). Videos can also be tracked using CPUs, with longer computational times.

**Animal rearing and handling.** For zebrafish videos we used the wild-type TU strain at 31 days post-fertilization (dpf). Animals were kept in 8-liter holding tanks at a density of ten fish per liter and with a 14-h light/10-h dark cycle in the main fish facility. For each experiment, a holding tank with the necessary number of fish was transported to the experimental room, where fish were carefully transferred to the experimental arena with a standard fish net appropriate for their age.

For the fruit fly videos, we used adults from the Canton S wild-type strain at 2–4 d post-eclosion. Animals were reared on a standard fly medium and kept on a 12-h light/dark cycle at 28 °C. We placed flies in the arena either by anesthetizing them with $CO_2$ or ice, or by using a suction tube. We found the latter method to have the least negative effect on the flies' health and to result in better activity levels.

**Experimental setups.** *Zebrafish video setup*. The main tank was placed inside a box built with matte white acrylic walls (Supplementary Fig. 3a). The lighting was based on infrared and RGB LED strips. A cylindrical retractable light diffuser made of plastic ensured homogeneous illumination in the central part of the main tank. A 20 MP monochrome camera (Emergent Vision HT-20000M) with a 28-mm lens (ZEISS Distagon T* 28-mm f/2.0 Lens with ZF.2) was positioned approximately 70 cm from the surface of the arena. To prevent reflections of the room ceiling, we used black fabric to cover the top of the box (Supplementary Fig. 3b). We used this setup to record zebrafish in groups and in isolation. Videos of groups of 10, 60 and 100 fish were recorded in a custom-made one-piece circular tank of 70-cm diameter, designed in-house. The tank was filled with fish system water to a depth of 2.5 cm. The circular tank was held in contact with the water of the main tank approximately 10 cm above a white background to improve the contrast between animals and background (Supplementary Fig. 3c). A water-recirculating system equipped with a filter and a chiller ensured a constant water temperature of 28 °C.

*Fruit fly video setup.* The setup was placed in a dedicated experimental room with controlled humidity (60%) and temperature (25 °C). RGB and IR LEDs placed on a ring around a cylindrical light diffuser guaranteed homogeneous light conditions in the central part of the setup (Supplementary Fig. 3a). Videos were recorded with the same camera as in the zebrafish setup. Black cardboard around the camera reduced reflections of the ceiling on the glass covering the arena (Supplementary Fig. 3b). We used two different arenas made of transparent acrylic, both built to prevent animals from walking on the walls. Arena 1 (diameter, 19 cm; height, 3 mm) had vertical walls that were heated with a white insulated resistance wire (Pelican Wire Company; 28 AWG solid (0.0126 inch), Nichrome 60, 4.4 Ω/ft, 0.015-inch white TFE tape). At 10 V, 0.3 A, the temperature at the walls reached 37 °C. Arena 2 (diameter, 19 cm; height, 3.4 mm) had conical walls (angle of inclination, 11°; width of conical ring, 18 mm) (Supplementary Fig. 3c). The best results were obtained with standard top-view recording (Supplementary Table 5). Arena 1 was also used for bottom-view recordings. The top of the arena was a sheet of glass covered with Sigmacote SL2 (Sigma-Aldrich), which prevented the flies from walking on the ceiling. A white plastic sheet was placed below the arena to increase the contrast between flies and background, at a distance of 5 cm below the arena to avoid shadows. To move flies into the arena, we either anesthetized them with $CO_2$ or ice, or used a suction tube. We found the latter method to have the least negative effect on the flies' health as evidenced by their activity levels.

**Individual image dataset.** *Individual image dataset setup*. We recorded 184 juvenile zebrafish (TU strain, 31 dpf) in separate chambers (60-mm-diameter Petri dishes). A holding grid with transparent acrylic walls allowed equal spacing between arenas while granting visual access to the neighboring dishes (Supplementary Fig. 2a). To increase image contrast, we used a white acrylic floor placed 5 cm below the holding grid, which acted as a light diffuser to prevent shadows. Four individuals at a time were recorded for 10 min (Supplementary Fig. 2b). On the outer borders we placed additional dishes containing fish to act as social stimuli (Supplementary Fig. 2b). This made the recorded fish swim more than they would have in isolation. From the 46 videos recorded, individual images were labeled according to the individual they represented. Each image was preprocessed according to the procedure detailed in the Supplementary Notes and then cropped as a square image for use in testing the identification network (image size, 52 × 52 pixels; Supplementary Fig. 2c). The dataset comprised a total of ~3,312,000 uncompressed, grayscale, labeled images.

**Statistics and reproducibility.** Results similar to the ones in Fig. 1c,g–j were obtained for $n = 32$ independent experiments tracked with idtracker.ai. Supplementary Tables 5–7 show the results of every experiment, and Supplementary Fig. 5 shows a comparison of the estimated accuracy and the accuracy after human validation.

In total, we performed $n = 13$ independent fish-fight experiments with different animals. We obtained results similar to those presented in Fig. 2a,b (Supplementary Fig. 10).

We tracked $n = 1$ video of ants (Fig. 2c–e) to illustrate the kind of graph analysis that can be performed with the trajectories obtained with idtracker.ai. We tracked $n = 3$ different videos of 100 juvenile zebrafish. We obtained results similar to those shown in Fig. 2f–h for the other two videos (Supplementary Fig. 11).

In Fig. 2e,g, $R^2$ was computed as the square of the Pearson's $r$ correlation coefficient, and $P$ is the two-sided $P$ value for a hypothesis test whose null hypothesis is that the slope is zero, obtained by Wald test with $t$-distribution of the test statistic.

**Artificial neural network details.** *Architectures.* The crossing-detector network (Fig. 1d) is a convolutional neural network[8,10]. It has two convolutional layers that obtain from data a relevant hierarchy of filters. A hidden layer of 100 neurons then transforms the convolutional output into a classification of 'single animal' or 'crossing'. idtracker.ai trains this network using images that can be confidently characterized as single or multiple animals (i.e., single animals as blobs consistent with single-animal statistics, and not split into more blobs in the past or future). Further details of the architecture are given in Supplementary Table 1.

The architecture of the identification network (Fig. 1e) consists of three convolutional layers, a hidden layer of 100 neurons and a classification layer with as many classes as animals in the collective. Further details are given in Supplementary Table 1. We tested variations of the architecture by modifying either the number of convolutional layers (Supplementary Table 2) or the number of hidden layer neurons (Supplementary Table 3). Analysis of these networks indicated that the most important requirement for successful identification is that the convolutional part have at least two layers (Supplementary Fig. 1). The GUI allows users to modify the architecture of this network and its training hyperparameters.

*Training.* The convolutional and fully connected layers of both networks are initialized via Xavier initialization[14]. Biases are initialized to 0.

The deep crossing-detector network is trained using the algorithm and hyperparameters described by Kingma and Ba[15]. The learning rate is set at the initial value of 0.005. This network is trained in mini batches of 100 images.

The identification network is trained using stochastic gradient descent, setting the learning rate to 0.005. This network is trained in mini batches of 500 images. In the training set, every image is duplicated via 180° rotation because the preprocessed images can have the head either in the upper-right corner or in the lower-left corner. Further details are given in the Supplementary Notes.

Overfitting was prevented by training with early stopping[16], that is, training the model until the error in the validation dataset reached a minimum. It is possible to use dropout[17] by modifying the GUI's advanced settings. These settings also allow transfer learning[11] using a network previously trained on other experiments.

**Heuristics.** *Classification of individual and crossing blobs.* The dataset used to train the convolutional crossing detector is extracted automatically from the video, following two heuristics that use properties of the blobs obtained during the preprocessing.

A blob $b$ is a collection of connected pixels in a given frame that do not belong to the background. First, a model of the area of individual blobs is constructed, considering the median number of pixels, $m_a$, and the s.d., $\sigma_a$, of the blobs in parts of the videos where the number of blobs corresponds to the number of animals declared by the user. A blob is considered to be a potential individual if its number of pixels differs from $m_a$ by less than $4\sigma_a$; otherwise it is categorized as a potential crossing.

Second, we consider the overlap of blobs in subsequent frames. We say that two blobs overlap in consecutive frames if the intersection of the set of pixels defining every blob is not empty. We classify the image corresponding to the blob $b$ as an individual if (i) $b$ is an individual as defined by the first heuristic, (ii) $b$ overlaps in the immediately previous and subsequent frames with only one blob, and (iii) every blob overlapping in the past and future of $b$ overlaps with at least one blob. We classify the images corresponding to the blob $b$ as crossings if (i) $b$ is a crossing as defined by the first heuristic and (ii) $b$ overlaps with more than one blob in the past or in the future. Or, (i) $b$ is a crossing as defined by the first heuristic and (ii) any blob overlapping in the past or in the future with $b$ overlaps with more than one blob. The Supplementary Notes include a formal definition of the heuristics.

*High-quality global fragment extraction.* During the protocol cascade, we use the identification network to assign all the global fragments whose images were not in the training dataset. Then, from these global fragments the algorithm extracts those with high quality, which will form part of the next training set. A global fragment is defined as being of high quality if it fulfills the three following

heuristics: First, the identity of all the individual fragments in the global fragment must be certain enough. Second, the identity of all the individual fragments in the global fragment must be consistent with the identity already assigned. Third, all the identities assigned to the individual fragments in the global fragment are unique. The Supplementary Notes include a formal definition of the heuristics.

**Image quality conditions.** *Resolution.* To test the performance of the system as a function of the number of pixels per animal, we artificially reduced the resolution of every frame, resizing it by a factor $\rho = [0.75, 0.5, 0.35, 0.25, 0.15]$. We reduced Moire effects by resampling using pixel area relation[18].

*Blurring.* To test the performance of the system under different levels of image blurring, we artificially blurred every frame, using a Gaussian kernel with s.d. $\sigma = [0.5, 1, 2, 3, 4, 5]$. The kernel size was computed automatically from the s.d.[18].

*Compression.* To test the robustness of the system under the effects of compression algorithms, we encoded the raw videos using the MPEG-4 and H.264 video codecs. Videos were encoded using FFmpeg. We encoded MPEG-4 by setting the FFmpeg's parameter qscale to 1. For the H.264 video codec, we used a constant rate factor of 0, which corresponds to a lossless compression.

*Inhomogeneous light conditions.* To test the robustness of the system under inhomogeneous light conditions, we switched off the infrared LEDs in two of the four walls and added a black cloth covering half of the cylindrical light diffuser (Supplementary Fig. 3). The videos recorded with the setups described in Supplementary Figs. 3 and 4 were obtained with IR illumination. Recording with this illumination allowed us to use any patterns of visible light that might have been needed in the experiment.

**Transfer learning.** The transfer-learning[11] technique can be applied at the identification stage to reuse knowledge from a network previously trained with similar animals and light conditions. In the advanced settings of the GUI, the user can apply transfer learning and decide whether to train the whole identification network or only the last two fully connected layers. We trained only the last two fully connected layers of the identification network when we applied this method.

**Parameters.** idtracker.ai needs the user to input parameters only for the segmentation of animals from the background. Users typically need to input only three parameters: a maximum intensity to separate animals from background, a minimum area to discard small objects during segmentation, and the number of animals to be tracked. In some cases, the user might need to input the minimum intensity or the maximum area, but we find this to be uncommon. The system then computes the number of animals in the video and asks the user for confirmation. There is also a set of advanced parameters that are divided in two main categories. It is possible to refine the preprocessing as detailed in http://idtracker.ai/quickstart.html, and to modify many of the identification network hyperparameters. By acting directly from the GUI, the user can define the presence and amount of dropout in the fully connected layers, choose an optimizer or choose one of the architectures described in Supplementary Tables 2 and 3. A detailed description of each parameter is provided in Supplementary Table 12 and in the explanation of the algorithm in the Supplementary Notes.

**Analysis of tracks.** *Attack score in zebrafish fights.* For each frame, we considered one focal animal to be attacking the other if its speed was greater than 1.5 body lengths (BL) per second, the other animal was positioned within an angle of $\pm 45°$ with respect to the focal animal's direction of movement, and the distance between them was less than 2 BL. We then calculated the attack score at time $t$ as the fraction of frames the focal animal spent attacking the other in the window $t \pm 1$ min.

*Interaction network in ants.* We represent the interaction network of a collective as a directed graph. An arrow between nodes $i$ and $k$ indicates that, during the video, individual $i$ triggered at least a locomotor response in $k$, and the thickness of that arrow is proportional to the number of triggered locomotor responses.

To detect interactions among individuals, we first computed an activity time series for each of them. We obtained the speed of each individual as the first derivative of the respective trajectory, which we smoothed in time with a moving average of window size $w_s$. The Hilbert envelope[19] of the smoothed individual speeds accentuated ramps in the speed time series. Finally, by applying the softmax function, we obtained the frame-wise probability mass function of an individual being active with respect to the collective as a measure of activity for each animal, $a_i(t)$. We then computed the local maxima of each $a_i(t)$. Two subsequent maxima in the activity of an individual are acceptable if they are at least $w_s$ frames apart. Let us consider the $i$th individual and call the frames corresponding to the local maxima of its activity, $M_i = \{m_{i,1}, \ldots, m_{i,n}\}$. For each acceptable local maximum $m_{i,1}$, we considered the activity of the other individuals in the frame interval $[m_{i,j}, m_{i,j} + w_t]$, where $w_t$ is the windows of frame in which the activity of an individual can be triggered by the focal individual. So, if another individual, say, $k$, reaches maximum activity within this interval and the distance between individuals $i$ and $k$ at frame $m_{i,j}$ is smaller than a fixed radius $r$, we say that $i$ triggered a locomotor response in $k$. We used $w_s = 59$ frames (1 s). $w_t = w_s$, and $r = 3$ BL.

*Location and average speed in milling groups.* We recorded and tracked three groups of 100 juvenile zebrafish while they were milling in a circular tank (Supplementary Fig. 3 presents the details of the setup). The trajectories were smoothed using a Gaussian kernel with an s.d. of two frames. We estimated the location of the center of the tank as the average center of mass over all the animals along the video. For every frame we computed the distance from each individual to the center of the tank. We used kernel density estimation[20] to estimate the probability density function of the location in the tank of three representative individuals: the one with highest average distance to the center in the video, the one with the smallest average distance to the center in the video, and a third individual with an intermediate average distance to the center (Fig. 2g).

The speed was computed as the norm of the velocity vector. Using standard linear regression analysis, we computed the correlation across individuals between the average distance to the center of the tank and the average speed (Fig. 2h).

**Reporting Summary.** Further information on experimental design is available in the Nature Research Reporting Summary linked to this article.

## Code availability

idtracker.ai is open-source and free software (license GPL v.3). The source code and the instructions for its installation are available at https://gitlab.com/polavieja_lab/idtrackerai. A quick-start user guide and a detailed explanation of the GUI can be found at http://idtracker.ai/. The software is also provided as Supplementary Software.

## Data availability

Processed data that can be used to reproduce all figures and tables can be found at http://idtracker.ai/. Lossless compressed videos can be downloaded from the same page. Raw videos are available from the corresponding author upon reasonable request. A library of single-individual zebrafish images for use in testing identification methods also can be found at http://idtracker.ai/. Two example videos, one of 8 adult zebrafish and one of 100 juvenile zebrafish, are also included as part of the quick-start user guide.

## References

13. Martins, S. et al. *Zebrafish* **13**, S47–S55 (2016).
14. Glorot, X. & Bengio, Y. in *Proc. Thirteenth International Conference on Artificial Intelligence and Statistics* (eds Teh, Y. W. & Titterington, M.) 249–256 (PMLR, Sardinia, Italy, 2010).
15. Kingma, D. & Ba, J. *arXiv* Preprint at https://arxiv.org/abs/1412.6980 (2015).
16. Morgan, N. & Bourlard, H. in *Advances in Neural Information Processing Systems 2* (ed Touretzky, D. S.) 630–637 (Morgan Kaufmann, San Francisco, 1990).
17. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014).
18. Bradski, G. *Dr. Dobb's Journal* **25**, 120–123 (2000).
19. Oppenheim, A. V. & Schafer, R. W. *Discrete-time Signal Processing* (Pearson, Upper Saddle River, NJ, 2014).
20. Scott, D. W. *Multivariate Density Estimation: Theory, Practice, and Visualization* (John Wiley & Sons, Hoboken, NJ, 2015).

nature methods

SUPPLEMENTARY INFORMATION

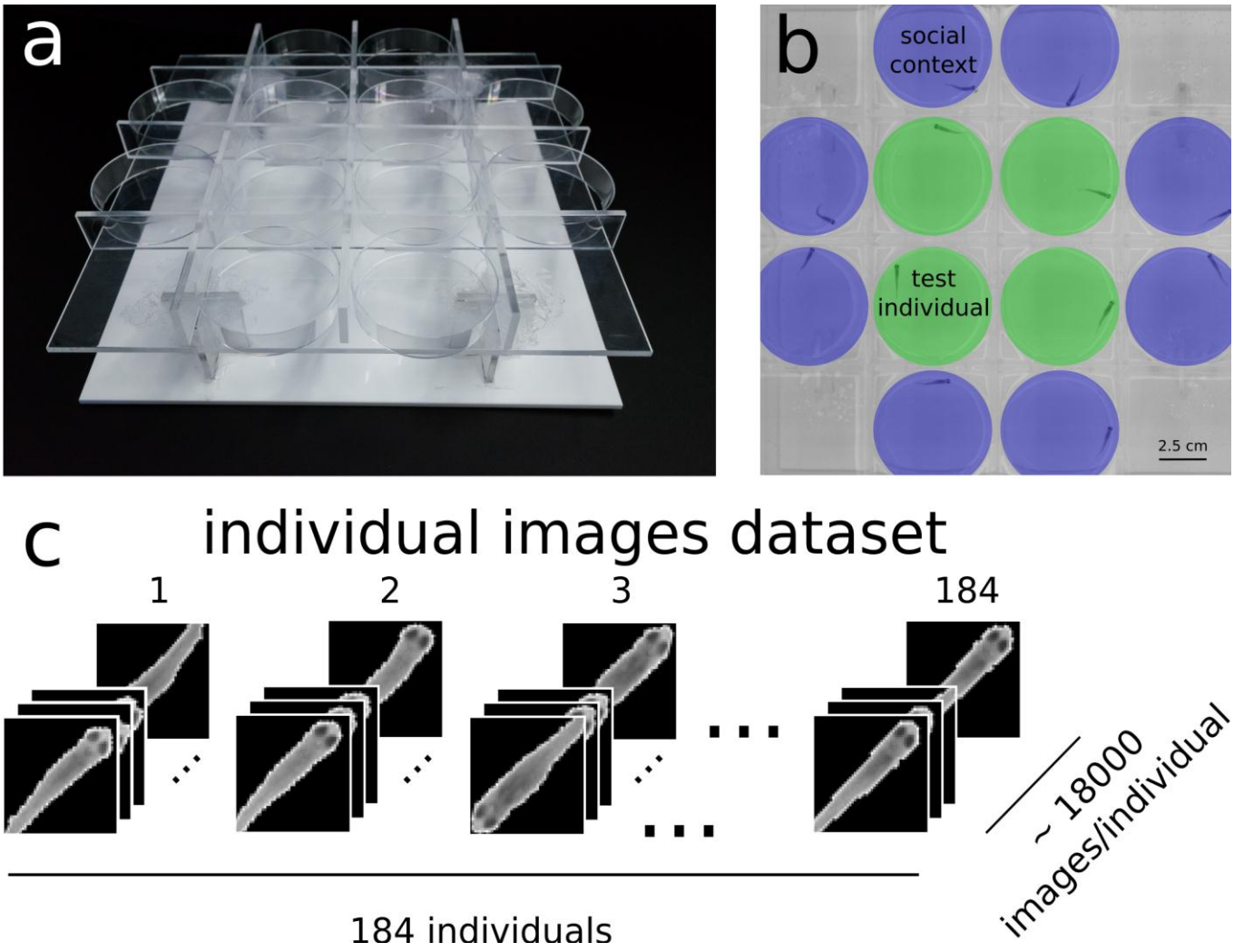
https://doi.org/10.1038/s41592-018-0295-5

In the format provided by the authors and unedited.

# idtracker.ai: tracking all individuals in small or large collectives of unmarked animals

**Francisco Romero-Ferrero[1,2], Mattia G. Bergomi[1,2], Robert C. Hinz[1], Francisco J. H. Heras[1] and Gonzalo G. de Polavieja[1]***

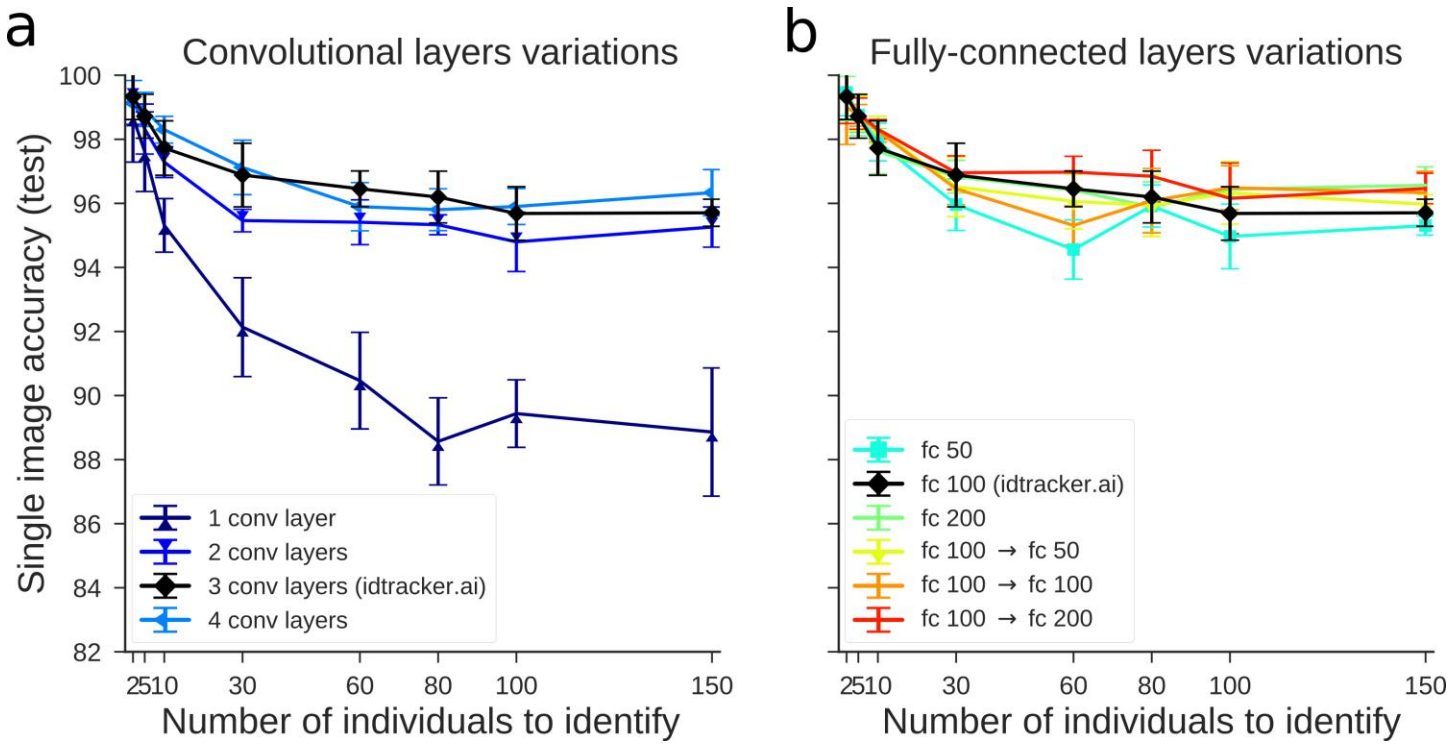
[1]Champalimaud Research, Champalimaud Center for the Unknown, Lisbon, Portugal. [2]These authors contributed equally: Francisco Romero-Ferrero, Mattia G. Bergomi. *e-mail: gonzalo.polavieja@neuro.fchampalimaud.org




**Supplementary Figure 1**

Training dataset of individual images.

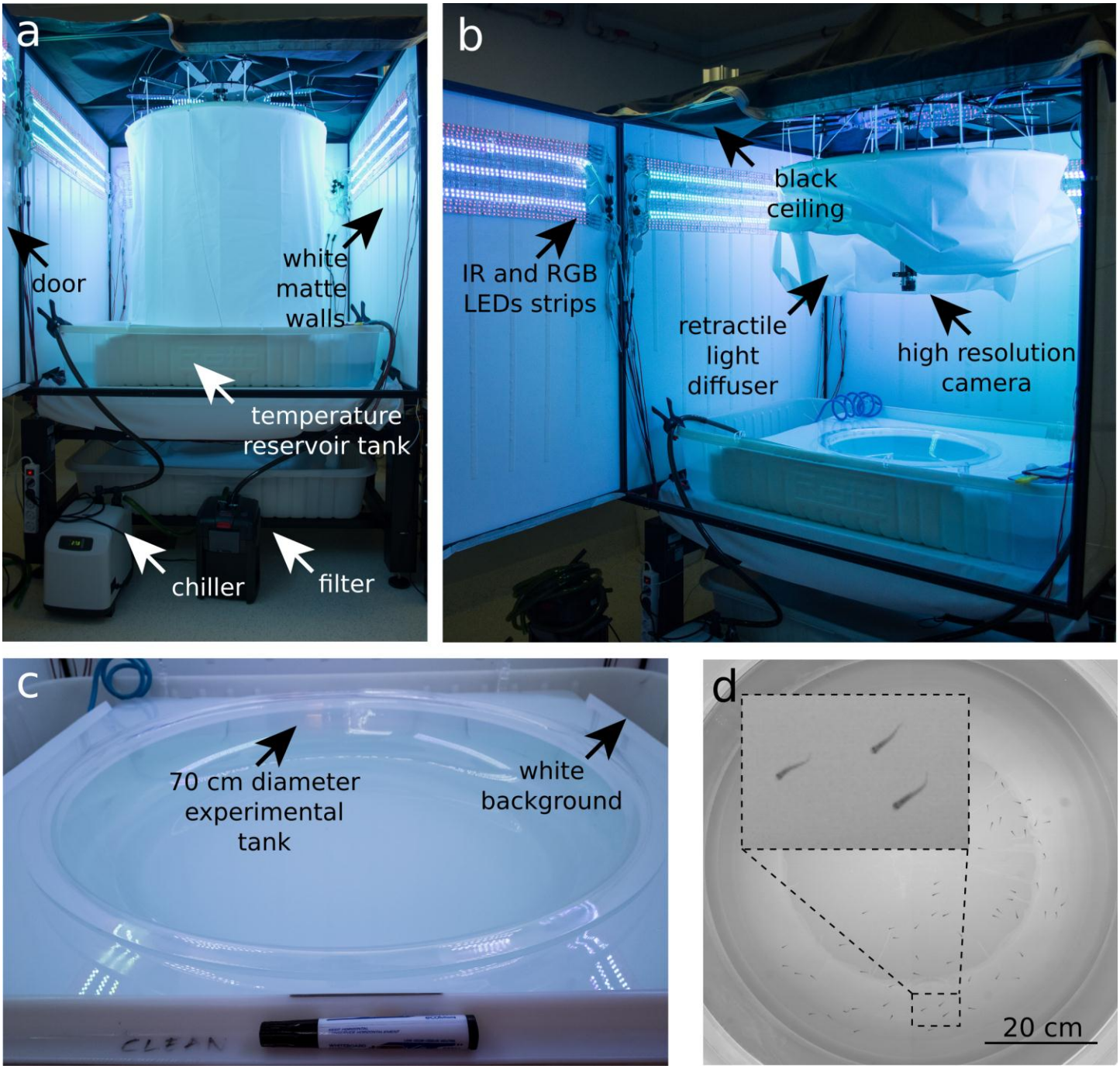
**(a)** Holding grid used to record 184 juvenile zebrafish (TU strain, 31 dpf) in separated chambers (60-mm-diameter Petri dishes). **(b)** Sample frame showing the individuals used to create the dataset and the individuals used as social context ($n$ = 46 videos corresponding to $n$ = 184 different individuals; ~18,000 frames per individual). **(c)** Summary of the individual-images dataset. The dataset is composed of a total of ~3,312,000 uncompressed, grayscale, labeled images (52 × 52 pixels).




**Supplementary Figure 2**

Single-image identification accuracy for different group sizes and different variations of the identification network.

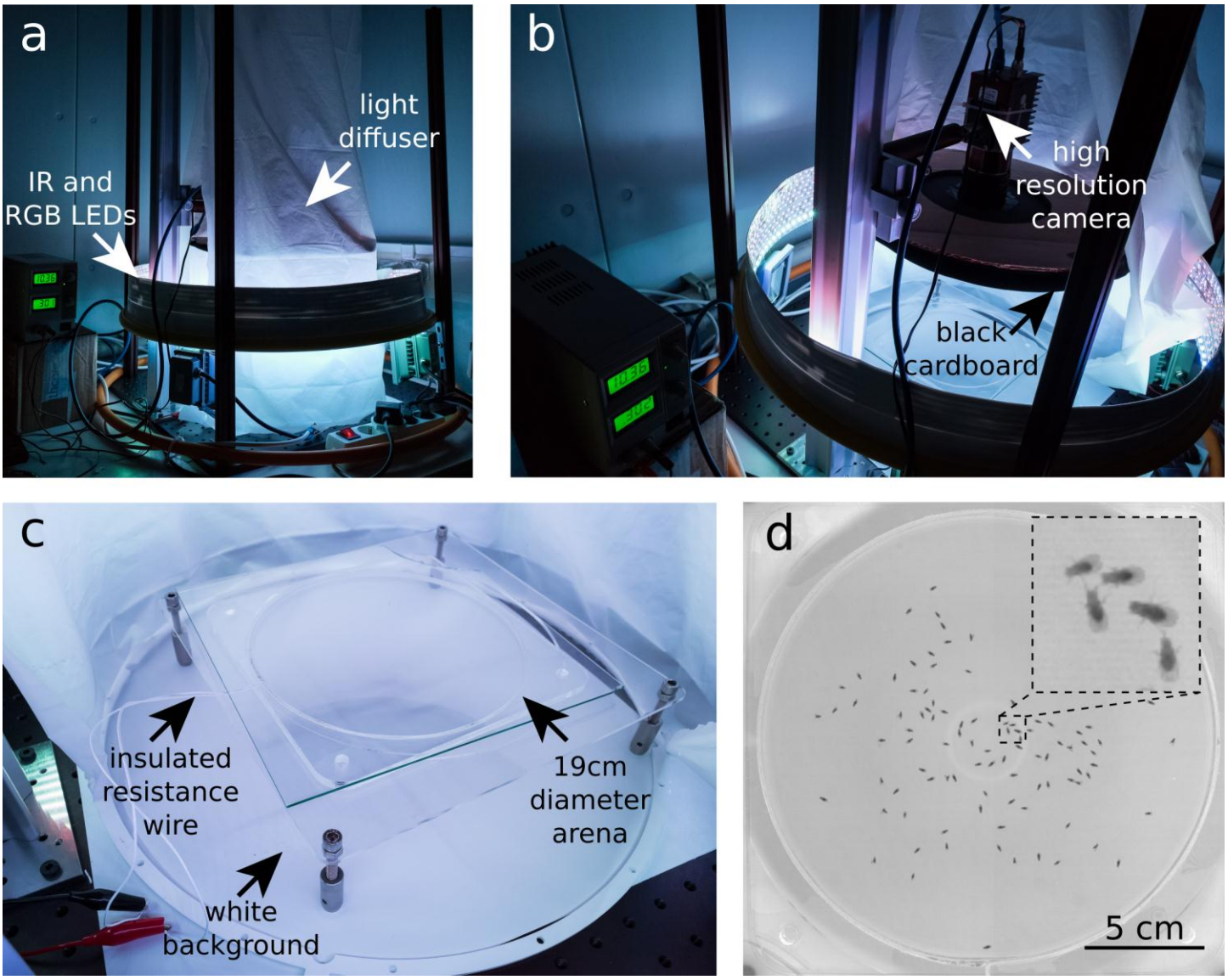
Each network is trained from scratch using 3,000 temporally uncorrelated images per animal (90% for training and 10% for validation) and then tested with 300 new temporally uncorrelated images to compute the single-image identification accuracy (Supplementary Notes). We train and test each network five times. For every repetition, the individuals of the group and the images of each individual are selected randomly. Images are extracted from videos of 184 different animals recorded in isolation (Supplementary Fig. 2). Colored lines with markers represent single-image accuracies (mean ± s.d., $n$ = 5) for network architectures with different numbers of convolutional layers (**a**; see Supplementary Table 2 for the architectures) and different sizes and numbers of fully connected layers (**b**; see Supplementary Table 3 for the architectures). The black solid line with diamond markers shows the accuracy for the network used to identify images in idtracker.ai (see Supplementary Table 1, identification convolutional neural network).




**Supplementary Figure 3**

Experimental setup for recording zebrafish videos.

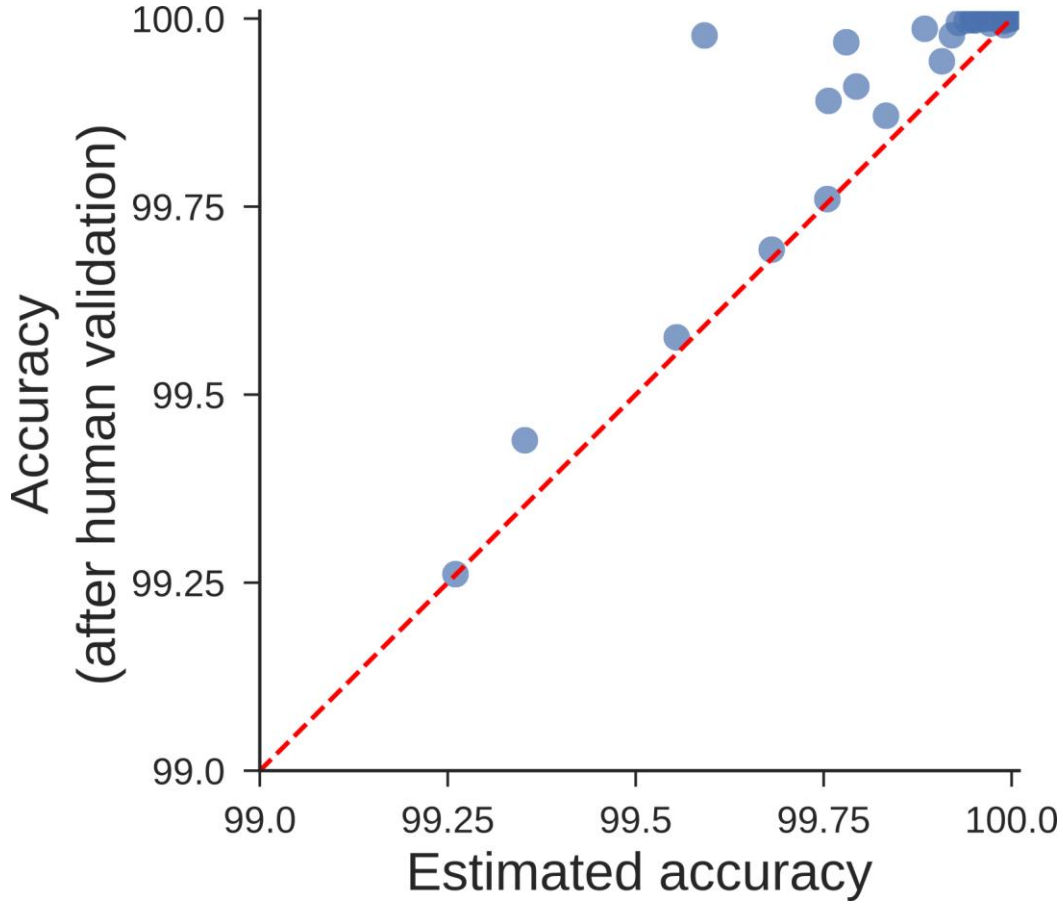
**(a)** Front view of the experimental setup used to record zebrafish in groups and in isolation. **(b)** Side view of the same setup with the light diffuser rolled up. **(c)** Close-up view of the custom-made circular tank used to record the groups of 10, 60 and 100 juvenile zebrafish. **(d)** Sample frame from a video of 60 animals ($n$ = 3 videos of 10 zebrafish, $n$ = 3 videos of 60 zebrafish, and $n$ = 3 videos of 100 zebrafish).




**Supplementary Figure 4**

Experimental setup used to record fruit fly videos.

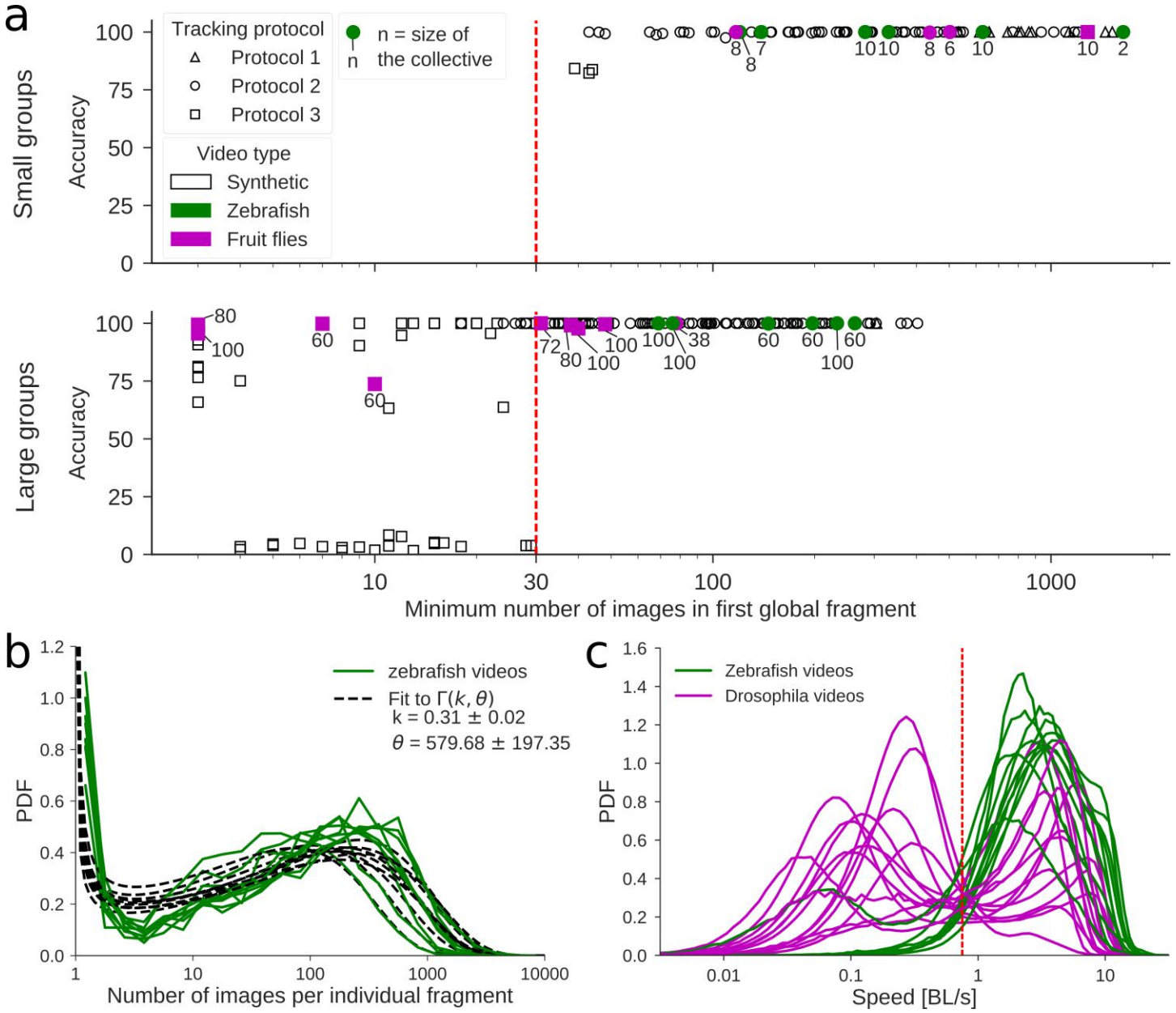
**(a)** Exterior view of the setup used to record flies in groups. **(b)** Top view of the same setup with the diffuser rolled up. **(c)** Close-up view of one of the two arenas used (arena 1). **(d)** Sample frame from a video of 100 flies ($n$ = 1 group of 38 flies, $n$ = 2 groups of 60 flies, $n$ = 1 group of 72 flies, $n$ = 2 groups of 80 flies, and $n$ = 3 groups of 100 flies; all animals were different for each group).




**Supplementary Figure 5**

Automatic estimation of identification accuracy.

Comparison between the accuracy estimated automatically by idtracker.ai and the accuracy computed by human validation of the videos (Supplementary Notes). The estimated accuracy is computed over the validated portion of the video. Blue dots represent the videos referenced in Supplementary Tables 5–7.

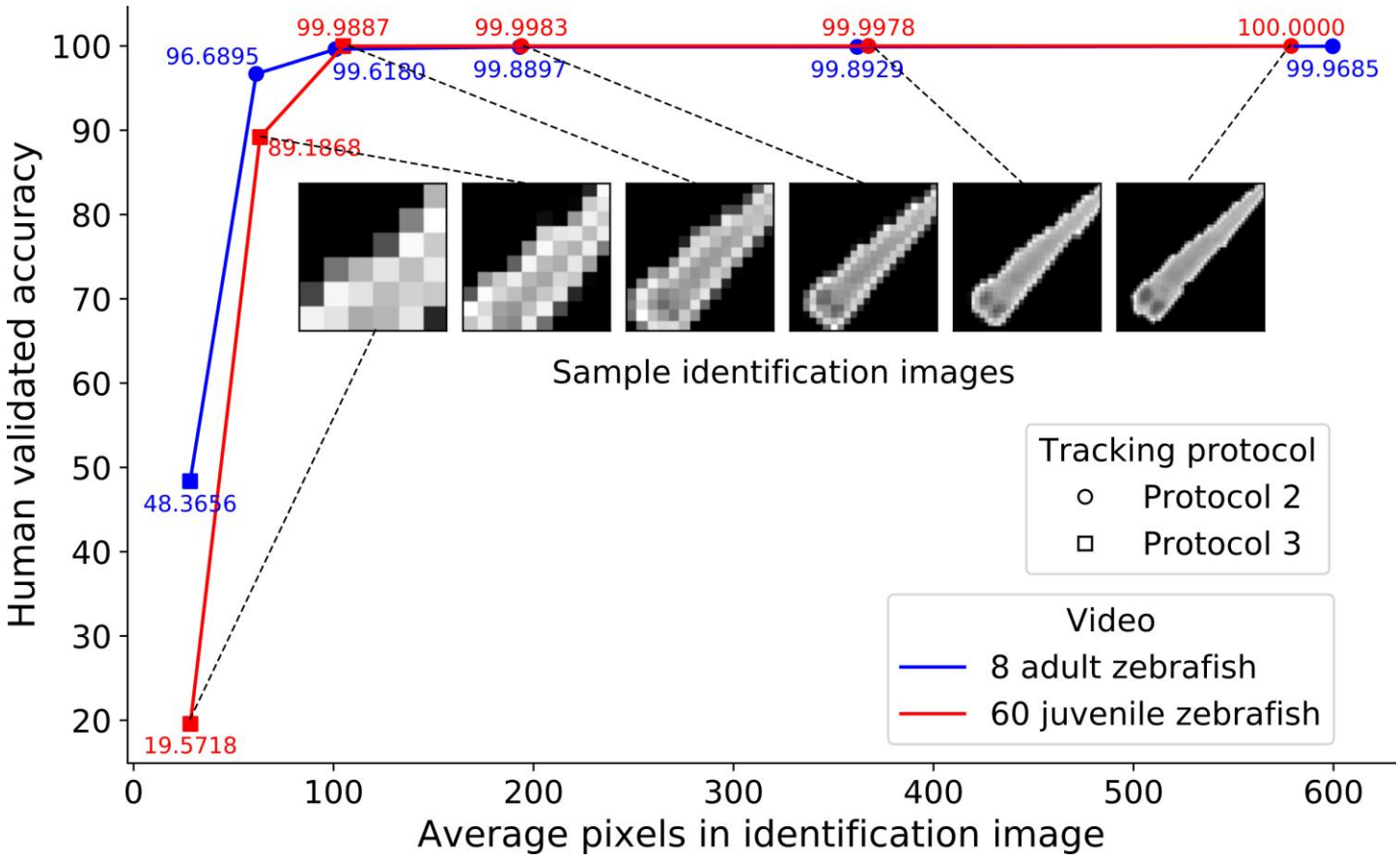



**Supplementary Figure 6**

Accuracy as a function of the minimum number of images in the first global fragment used for training.

To study the effect of the minimum number of images per individual in the first global fragment used to train the identification network, we created synthetic videos using images of 184 individuals recorded in isolation (Supplementary Fig. 1). Each synthetic video consists of 10,000 frames, where the number of images in every individual fragment was drawn from a gamma distribution, and the crossing fragments lasted for three frames (Supplementary Notes). The parameters were set as follows: $\theta$ = [2,000, 1,000, 500, 250, 100], $k$ = [0.5, 0.35, 0.25, 0.15, 0.05], number of individuals = [10, 60, 100]. For every combination of these parameters we ran three repetitions. In total, we computed both the cascade of training and identification protocols and the residual identification for 225 synthetic videos. **(a)** Identification accuracy for simulated (empty markers) and real videos (color markers) as a function of the minimum number of images in the first global fragment. The number next to each color marker indicates the number of animals in the video. The accuracy of the real videos was obtained by manual validation (Supplementary Tables 5–7). In some videos, animals are almost immobile for long periods of time because of low-humidity conditions. Potentially, the individual fragments acquired during these periods encode less information that is useful for identifying the animals. To account for this, we corrected the number of images in the individual fragments by considering only frames in which the animals were moving with a speed of at least 0.75 BL/s. We observed that idtracker.ai was more likely to have higher accuracy when the minimum number of images in the first global fragment used for training was >30. **(b)**

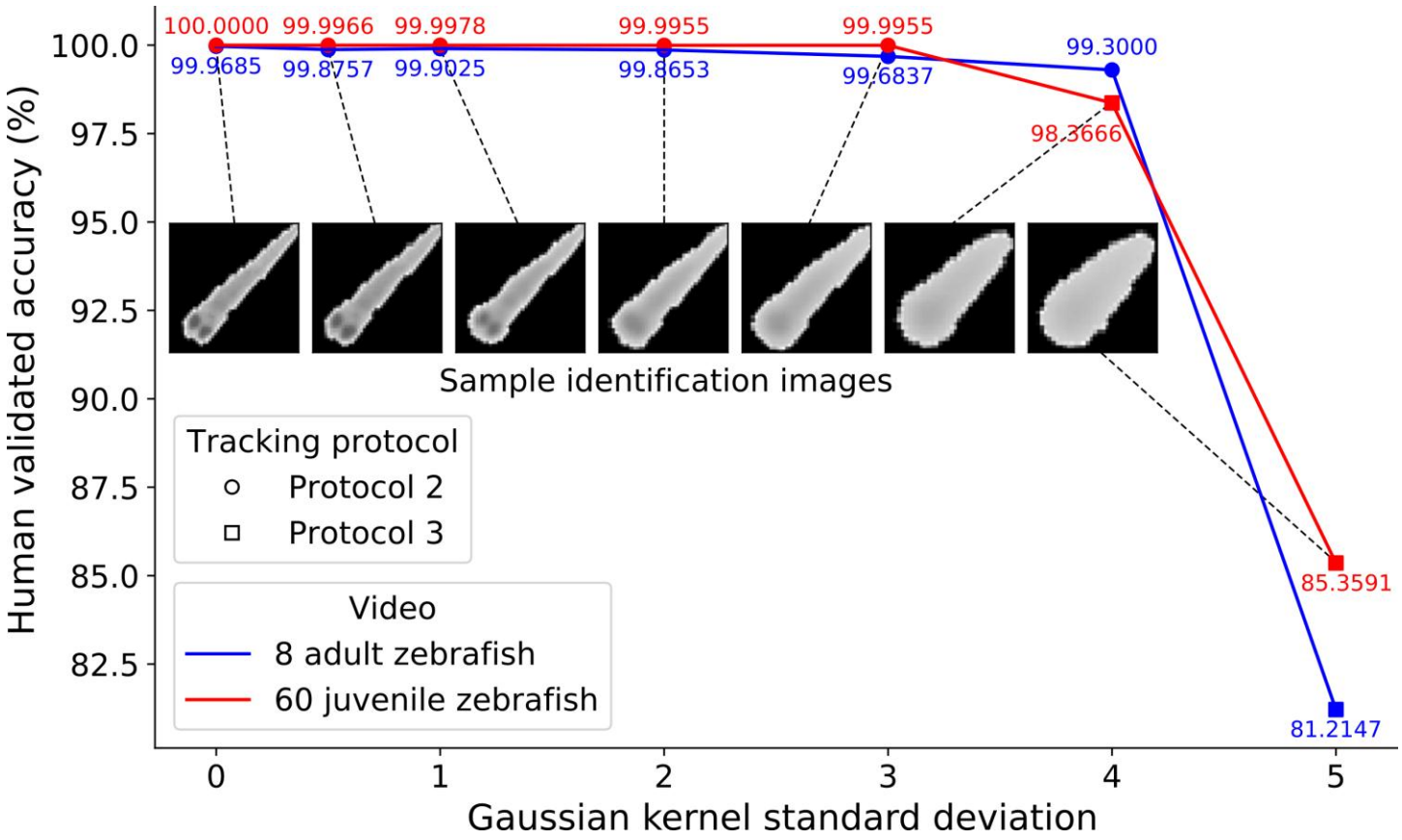
Distributions of the number of images per individual fragment for real videos of zebrafish, and their fits to a gamma distribution. **(c)** Distributions of speeds of zebrafish and fruit fly videos.




**Supplementary Figure 7**

Performance as a function of resolution.

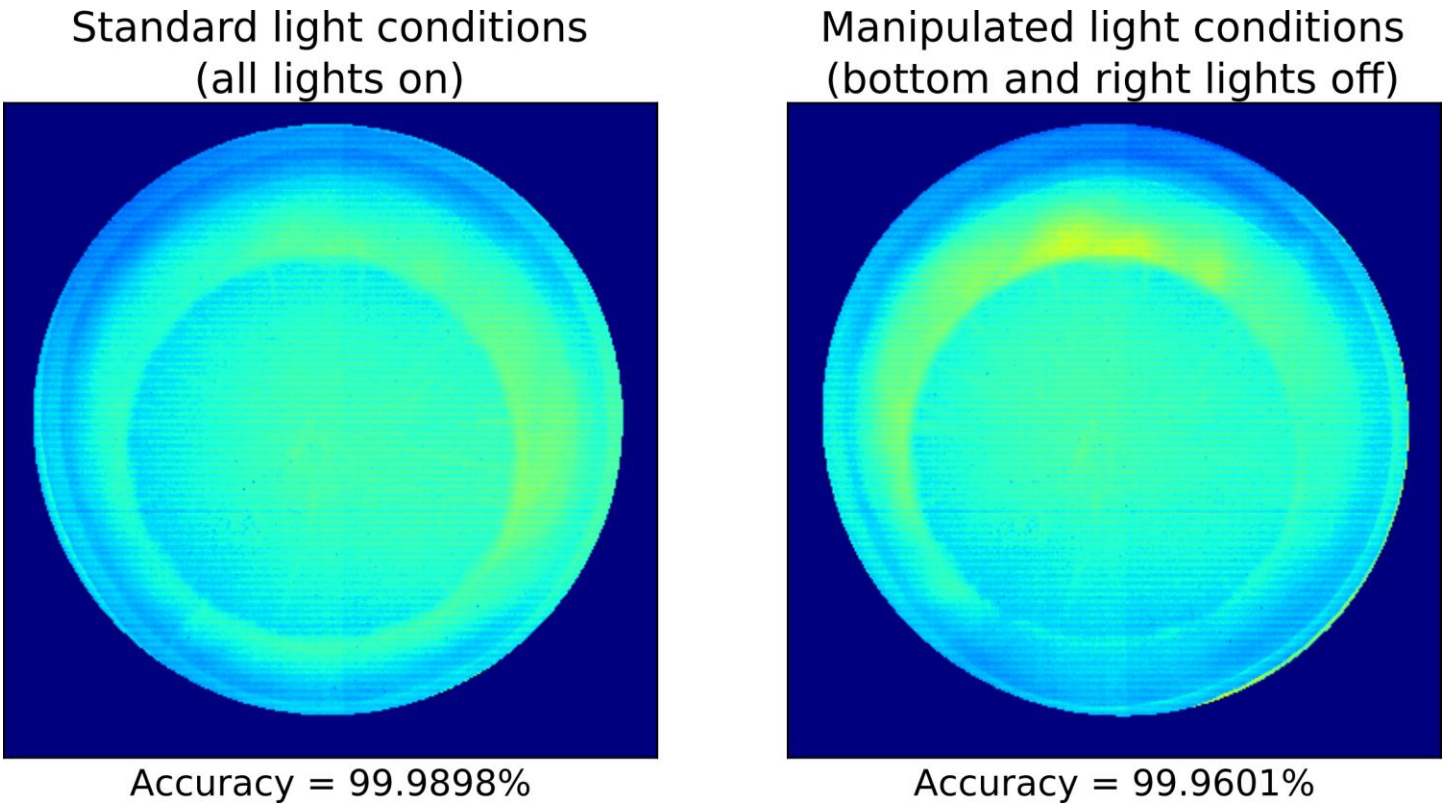
Human-validated accuracy of tracking results obtained at six different resolutions. Pixels per animal are here indicated at the identification stage. There are fewer pixels per animal at the segmentation stage—approximately 25 and 300 pixels per animal, compared with 100 and 600 at the identification stage, respectively.




**Supplementary Figure 8**

Performance after application of Gaussian blurring.

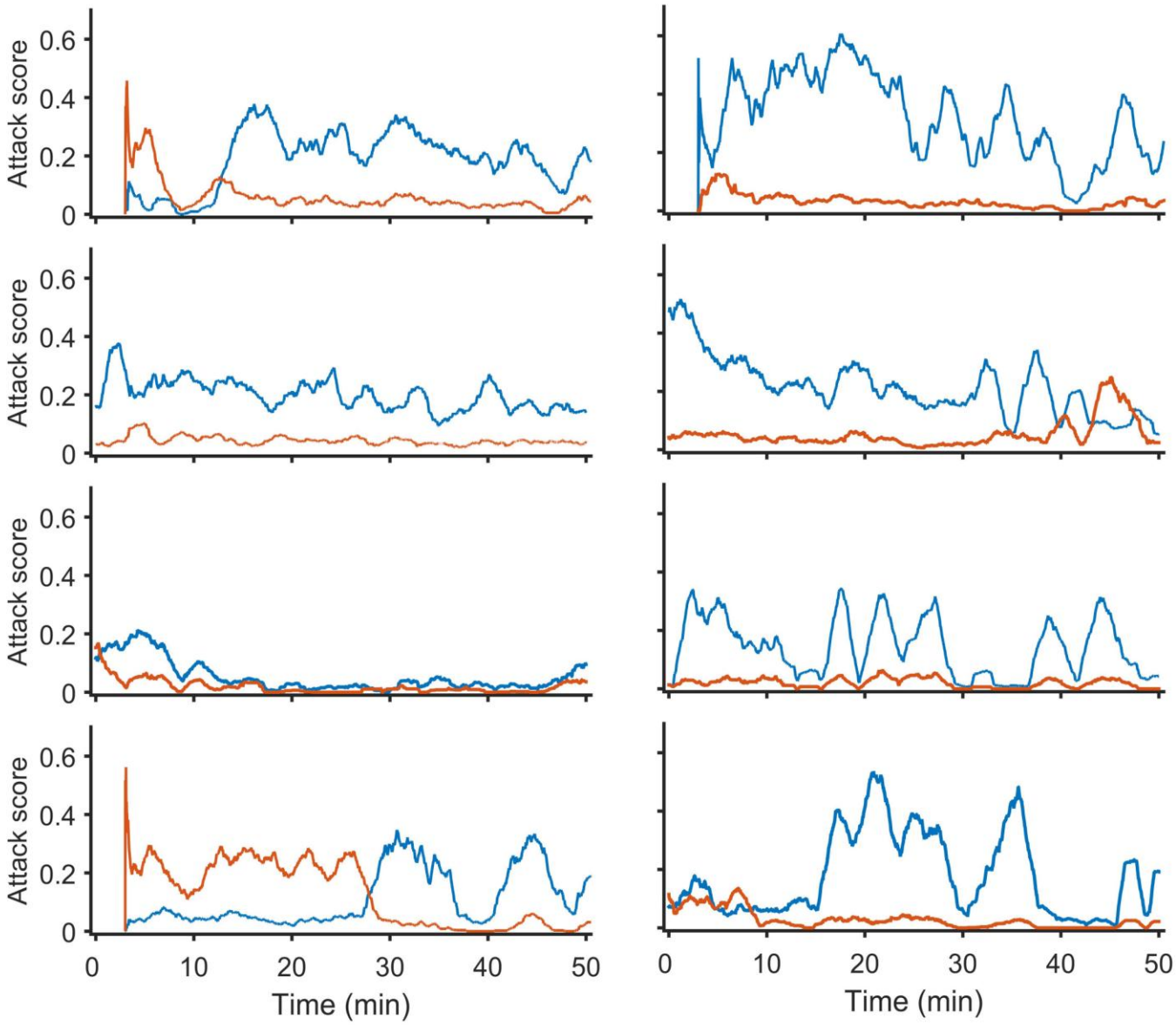
Human-validated accuracy of tracking results obtained at seven different values of the s.d. of a Gaussian filtering of the video.




**Supplementary Figure 9**

Performance with inhomogeneous light conditions.

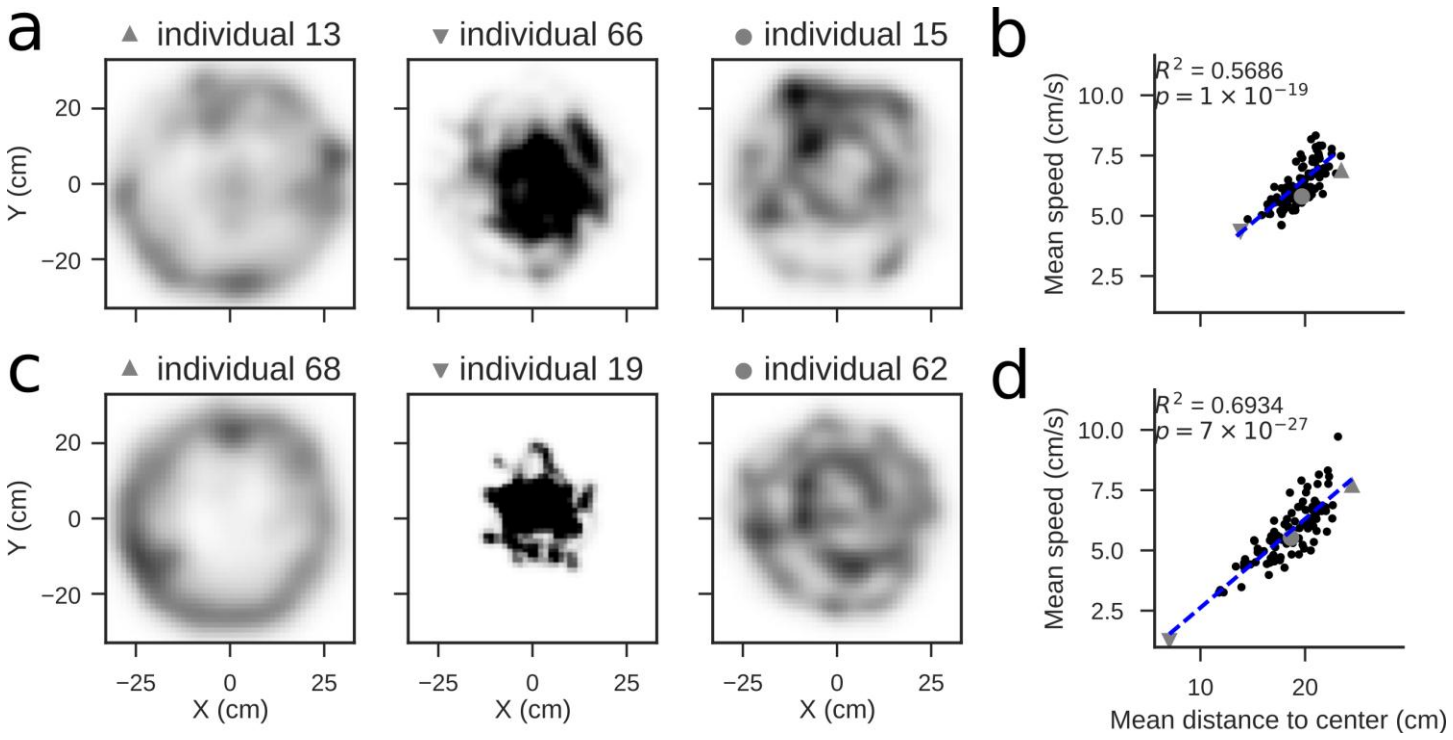
Background image corresponding to two different experiments with 60 zebrafish (*n* = 1 experiment for each condition). On the left for our standard setup and on the right after switching off the IR LEDs in two walls and covering the light diffuser in the same side with a black cloth. Human-validated accuracy of tracking results is given below the images. The background image is computed as the average of equally spaced frames along the video with a period of 100 frames.




**Supplementary Figure 10**

Attack score over time for seven pairs of fish staged to fight.

Each colored line represents the attack score of an individual (see the Methods for the definition of ‘attack score’).




**Supplementary Figure 11**

Correlation between the average distance to the center of the tank and the average speed for two milling groups of 100 juvenile zebrafish.

**(a)** Probability density of the location in the tank of three representative individuals depicted in (b) as gray markers. **(b)** Average speed along the video as a function of the average distance to the center of the tank for all the fish in the group. Each black dot represents an individual; the gray markers are the individuals depicted in (a). The blue dashed line is the line of best fit to the data ($R^2$ = 0.5686, Pearson's *r* and $P = 10^{-19}$, two-sided *P* value using Wald test with *t*-distribution of the test statistic). **(c)** Same as in (a) for a different video. **(d)** Same as in (b) for a different video ($R^2$ = 0.6934, Pearson's *r* and $P = 7 \times 10^{-27}$, two-sided *P* value using Wald test with *t*-distribution of the test statistic).

# Appendix A

# Supplementary Tables

**Supplementary Table 1: Convolutional neural networks used in idtracker.ai.** The crossing detector network is used to classify images as belonging to a single animal or to multiple animals crossing or touching. The identification network is used to identify individual images.

*Crossing detector network*

| Layer | operation | number of units | kernel size | stride |
|---|---|---|---|---|
| 1 | convolution - ReLu | 16 | (5,5) | 1 |
| 2 | max pooling | - | - | (2,2) |
| 3 | convolution - ReLu | 64 | (5,5) | 1 |
| 4 | max pooling | - | - | (2,2) |
| 5 | fully connected - ReLu | 100 | - | - |
| 6 | fully connected - softmax | 2 | - | - |

*Identification network*

| Layer | operation | number of units | kernel size | stride |
|---|---|---|---|---|
| 1 | convolution - ReLu | 16 | (5,5) | 1 |
| 2 | max pooling | - | - | (2,2) |
| 3 | convolution - ReLu | 64 | (5,5) | 1 |
| 4 | max pooling | - | - | (2,2) |
| 5 | convolution - ReLu | 100 | (5,5) | 1 |
| 6 | fully connected - ReLu | 100 | - | - |
| 7 | fully connected - softmax | group size | - | - |

**Supplementary Table 2: Architectures with variations in the number of convolutional layers part**. Several architectures with variable numbers and shape of convolutional layers has been tested in order to assess the stability of the accuracy in single-image identification.

***1 convolutional layer***

| Layer | operation | number of units | kernel size | stride |
|---|---|---|---|---|
| 1 | convolution - ReLu | 16 | (5,5) | 1 |
| 2 | fully connected - ReLu | 100 | - | - |
| 3 | fully connected - softmax | group size | - | - |

***2 convolutional layers***

| Layer | operation | number of units | kernel size | stride |
|---|---|---|---|---|
| 1 | convolution - ReLu | 16 | (5,5) | 1 |
| 2 | max pooling | - | - | (2,2) |
| 3 | convolution - ReLu | 64 | (5,5) | 1 |
| 4 | fully connected - ReLu | 100 | - | - |
| 5 | fully connected - softmax | group size | - | - |

***4 convolutional layers***

| Layer | operation | number of units | kernel size | stride |
|---|---|---|---|---|
| 1 | convolution - ReLu | 16 | (5,5) | 1 |
| 2 | max pooling | - | - | (2,2) |
| 3 | convolution - ReLu | 64 | (5,5) | 1 |
| 4 | max pooling | - | - | (2,2) |
| 5 | convolution - ReLu | 100 | (5,5) | 1 |
| 6 | max pooling | - | - | (2,2) |
| 7 | convolution - ReLu | 100 | (5,5) | 1 |
| 8 | fully connected - ReLu | 100 | - | - |
| 9 | fully connected - softmax | group size | - | - |

**Supplementary Table 3: Architectures with variations in the number and size of the fully connected layers (fc)**. Several architectures with variable numbers and shape of fully-connected layers has been tested in order to assess the stability of the accuracy in single-image identification. The notation *fc* $n \rightarrow$ *fc* $m$ characterises each architecture according to its fully-connected layers before the output layer.

***fc 50 units***

| Layer | operation | number of units | kernel size | stride |
|---|---|---|---|---|
| 1 | convolution - ReLu | 16 | (5,5) | 1 |
| 2 | max pooling | - | - | (2,2) |
| 3 | convolution - ReLu | 64 | (5,5) | 1 |
| 4 | max pooling | - | - | (2,2) |
| 5 | convolution - ReLu | 100 | (5,5) | 1 |
| 6 | fully connected - ReLu | 50 | - | - |
| 7 | fully connected - softmax | group size | - | - |

***fc 200 units***

| Layer | operation | number of units | kernel size | stride |
|---|---|---|---|---|
| 1-5 | (see first model) | | | |
| 6 | fully connected - ReLu | 200 | - | - |
| 7 | fully connected - softmax | group size | - | - |

***fc 100 units → fc 50 units***

| Layer | operation | number of units | kernel size | stride |
|---|---|---|---|---|
| 1-5 | (see first model) | | | |
| 6 | fully connected - ReLu | 100 | - | - |
| 7 | fully connected - softmax | 50 | - | - |
| 8 | fully connected - ReLu | group size | - | - |

***fc 100 units → fc 100 units***

| Layer | operation | number of units | kernel size | stride |
|---|---|---|---|---|
| 1-5 | (see first model) | | | |
| 6 | fully connected - ReLu | 100 | - | - |
| 7 | fully connected - softmax | 100 | - | - |
| 8 | fully connected - ReLu | group size | - | - |

***fc 100 units → fc 200 units***

| Layer | operation | number of units | kernel size | stride |
|---|---|---|---|---|
| 1-5 | (see first model) | | | |
| 6 | fully connected - ReLu | 100 | - | - |
| 7 | fully connected - softmax | 200 | - | - |
| 8 | fully connected - ReLu | group size | - | - |

| Video | Condition | Protocol | Preprocess. time | Protocol 1 time/acc. | Protocol 2 time/acc. | Protocol 3 time/acc. | Postprocessing time/acc. | Total time |
|---|---|---|---|---|---|---|---|---|
| 8 zebrafish | raw | 2 | 0:04:18 | 0:01:09/98.110 | 0:15:14/99.490 | -/- | 0:01:48/99.891 | 0:22:31 |
| | G. blurring (3) | 2 | 0:04:40 | 0:02:22/90.514 | 0:16:31/98.876 | -/- | 0:01:42/99.870 | 0:25:17 |
| | G. blurring (6) | 2 | 0:04:57 | 0:01:30/ 52.649 | 0:16:25/96.772 | -/- | 0:02:06/99.191 | 0:24:59 |
| | Res. reducc. (3) | 2 | 0:03:40 | 0:01:16/96.454 | 0:14:19/99.513 | -/- | 0:01:45/99.907 | 0:21:02 |
| | Res. reducc. (5) | 2 | 0:02:32 | 0:01:30/95.182 | 0:08:40/ 98.163 | -/- | 0:01:36/99.526 | 0:14:20 |
| 60 zebrafish | raw | 2 | 3:11:17 | 0:10:09/95.494 | 1:37:34/99.982 | -/- | 0:22:50/99.997 | 5:21:52 |
| | G. blurring (4) | 2 | 6:58:23 | 0:22:12/94.828 | 1:49:25/99.821 | -/- | 0:22:15/99.995 | 9:32:16 |
| | G. blurring (6) | 3 | 3:54:03 | 0:12:46/33.890 | 0:05:48/33.890 | 15:20:18/89.787 | 1:29:28/90.629 | 21:02:24 |
| | Res. reducc. (2) | 2 | 3:19:34 | 0:11:36/93.126 | 1:44:13/99.996 | -/- | 0:20:24/99.998 | 5:35:49 |
| | Res. reducc. (4) | 3 | 2:15:41 | 0:11:51/65.643 | 0:04:41/65.643 | 14:50:19/99.315 | 0:31:55/99.991 | 17:54:28 |
| 100 zebrafish (3) | raw | 2 | 6:39:32 | 0:26:30/98.516 | 4:53:46/99.948 | -/- | 1:19:47/99.978 | 13:19:37 |

**Supplementary Table 4: Computational times and accuracies at different stages in the algorithm**. We show results for videos of 8, 60 and 100 zebrafish. The videos of 8 and 60 animals are the same videos as in Supplementary Tables 8 and 9, the numbers in brackets indicate the row number in the corresponding table. The video of 100 fish is the same as the third video in Supplementary Table 6. Differences in times and accuracy are due to the stochasticity introduced by the training algorithm (stochastic gradient descent) and due to differences in the preprocessing parameters. Preprocessing times include the segmentation, crossing detection and fragmentation. Protocol 1 times include the first quality check. Protocol 2 times include the second quality test. Protocol 3 times include the pretraining and accumulation steps of the protocol and the quality checks. Postprocessing times include the the residual identification, computation of the estimated accuracy, crossing solving and the generation of the trajectories. Accuracies of the protocols are computed from the individual fragments identities assigned according to eq. (B.2.7).

| Video | Duration | Frames per second | Pixels per animal (segmentation) | Protocol | Reviewed crossings | Accuracy prot. cascade | **Accuracy** | % Non-identified | % Misidentified |
|---|---|---|---|---|---|---|---|---|---|
| 2 nacre zebrafish (1) † | 15'58" | 24 | 764 | 2 | 90 | 100 | **99.991** | 0 | 0.009 |
| 7 inbred WIK zebrafish † | 08'19" | 31 | 670 | 2 | 393 | 100 | **99.997** | 0.003 | 0 |
| 8 zebrafish † ∗ | 05'22" | 28 | 331 | 2 | 842 | 100 | **99.969** | 0.028 | 0.003 |
| 10 zebrafish | 10'03" | 32 | 528 | 2 | 311 | 100 | **100** | 0 | 0 |
| 10 zebrafish | 10'03" | 32 | 551 | 2 | 877 | 100 | **99.999** | 0 | 0.001 |
| 10 zebrafish | 10'10" | 32 | 534 | 2 | 667 | 100 | **100** | 0 | 0 |
| 5 medaka fish † | 21'41" | 28 | 232 | 2 | 124 | 100 | **100** | 0 | 0 |
| 10 medaka fish † | 12'20" | 28 | 199 | 2 | 89 | 100 | **100** | 0 | 0 |
| 20 medaka fish † | 08'20" | 29 | 273 | 2 | 597 | 100 | **99.999** | 0.001 | 0 |
| 6 Drosophila (4 fem, 2 mal) † | 07'57" | 42 | 1048 | 2 | 306 | 100 | **99.994** | 0.006 | 0 |
| 8 female Drosophila (1) † | 03'07" | 28 | 819 | 2 | 199 | 100 | **99.986** | 0 | 0.014 |
| 8 female Drosophila (2) † | 14'58" | 28 | 750 | 2 | 298 | 99.882 | **99.760** | 0.104 | 0.136 |
| 2 agouti mice † | 05'19" | 49 | 12855 | 2 | 93 | 99.932 | **99.910** | 0 | 0.090 |
| 2 black mice (1) † | 12'24" | 49 | 11986 | 2 | 200 | 99.582 | **99.576** | 0 | 0.424 |
| 2 black mice (2) † | 07'06" | 49 | 11401 | 2 | 41 | 99.738 | **99.693** | 0 | 0.307 |
| 2 black mice (3) † | 07'06" | 49 | 12124 | 2 | 44 | 100 | **100** | 0 | 0 |
| 4 black mice (2)† | 33'58" | 25 | 7398 | 2 | 351 | 100 | **99.993** | 0.002 | 0.005 |
| 4 black mice (3)† | 52'54" | 24 | 6440 | 2 | 287 | 97.974 | **96.641** | 0.032 | 3.327 |

**Supplementary Table 5: Results of manual validation for video of small group size**. To compare the performance of idtracker.ai with respect to idTracker, we tracked and manually validated most of the videos used in [1]. We also add three more videos of 10 zebrafish (TU strain, 31 dpf). We observe that the performance is comparable to the one obtained by idTracker (see Supplementary Table 1 in [1]). The column "Reviewed crossings" displays the number of *crossing fragments* as defined in section B.2.3 in the validated part. The column "Accuracy prot. cascade" displays the proportion of individual images correctly identified after the protocol cascade. The column "Accuracy" displays the proportion of individual images correctly identified in the validated part. The column "Non-identified" displays the percentage of individual images for which the system did not assign an identity. The column "Misidentified" displays the percentage of individual images wrongly identified.

**Table legend:** ∗: used to develop the tracking system. † video from [1].

| Video | Duration | Frames per second | Pixels per animal (segmentation) | Protocol | Reviewed crossings | Accuracy prot. cascade | **Accuracy** / Indiv. acc. | % Non-identified | % Misidentified |
|---|---|---|---|---|---|---|---|---|---|
| 60 zebrafish | 10'29" | 31 | 325 | 2 | 338 | 99.921 | **99.871** / 99.880 | 0.024 | 0.106 |
| 60 zebrafish | 10'30" | 31 | 308 | 2 | 277 | 100 | **99.998** / 99.999 | 0.002 | 0 |
| 60 zebrafish | 10'09" | 32 | 303 | 2 | 213 | 100 | **100** / 99.999 | 0 | 0 |
| 100 zebrafish | 10'16" | 32 | 316 | 2 | 2552 | 100 | **99.977** / 99.989 | 0.016 | 0.007 |
| 100 zebrafish | 10'12" | 32 | 273 | 2 | 750 | 100 | **99.999** / 99.999 | 0.001 | 0 |
| 100 zebrafish ∗ | 10'10" | 32 | 309 | 2 | 628 | 100 | **99.997** / 100 | 0.002 | 0.001 |
| 38 fruit flies ↓ (2) | 10'07" | 36 | 602 | 2 | 753 | 99.996 | **99.978** / - | 0.002 | 0.020 |
| 60 fruit flies ♀↑ (1) | 10'00" | 51 | 372 | 3 | 699 | 100 | **99.999** / - | 0.001 | 0 |
| 72 fruit flies ↓ (2) | 10'00" | 36 | 602 | 3 | 165 | 100 | **99.997** / - | 0.003 | 0 |
| 80 fruit flies ↓ (1) | 10'02" | 34 | 589 | 3 | 514 | 99.925 | **99.439** / - | 0.420 | 0.141 |
| 80 fruit flies ↑ (2) | 10'02" | 51 | 273 | 3 | 222 | 100 | **99.891** / 99.046 | 0.109 | 0 |
| 100 fruit flies ↓ (1) | 10'00" | 28 | 558 | 3 | 107 | 99.909 | **99.580** / - | 0.327 | 0.093 |



**Supplementary Table 6: Results of manual validation for videos of large group size**. To validate the system we manually reviewed the identities of the animals before and after every crossing image and every non-identified image for a part of the video (see Global Validation in section B.3.1). For some videos we also reviewed the entire video for 10 randomly chosen animals (see Individual Validation in section B.3.1). "Reviewed crossings", "Accuracy", "Accuracy prot. cascade", "Non-identified" and "Misidentified" refer to the global validation, and "Indiv. acc."' refers to the individual validation. The column "Reviewed crossings" displays the number of *crossing fragments* as defined in section B.2.3 in the validated part. The column "Accuracy prot. cascade" displays the proportion of individual images correctly identified (PIICI) after the protocol cascade. The column "Accuracy" displays the PIICI in the validated part. The column "Indiv. acc." displays the average PIICI for the 10 validated individuals. The column "Non-identified" displays the percentage of individual images for which the system did not give an identity. The column "Misidentified" displays the percentage of individual images wrongly identified. All zebrafish videos were recorded in the setup described in Supplementary Figure 3. All the fruit flies videos were recorded in the setup described in Supplementary Figure 4.

**Table legend:** ∗: used to develop the tracking system and video in **Fig 1.** in main text. (1)/(2): video recorded in arena 1 or 2 respectively (see Supplementary Figure 4). ↑ or ↓: recorded from below (↑) or from the top (↓) respectively. ♀: the video realised with only female fruit flies.

| Video | Duration | Frames per second | Pixels per animal (segmentation) | Protocol | Reviewed crossings | Accuracy prot. cascade | **Accuracy** | % Non-identified | % Misidentified |
|---|---|---|---|---|---|---|---|---|---|
| **Compressed videos with bad illumination conditions (users reported difficulties using idTracker [1])** | | | | | | | | | |
| 10 fruit flies | 10'12" | 60 | 420 | 3 | 288 | 100 | **100** | 0 | 0 |
| 14 ants | 13'10" | 59 | 573 | 2 | 498 | 99.989 | **99.943** | 0.001 | 0.056 |
| **Very low locomotor activity levels for most animals or dead animals (low humidity)** | | | | | | | | | |
| 60 fruit flies | 10'14" | 51 | 292 | 3 | - | - | 73.720 † | - | - |
| 100 fruit flies ♀ | 10'14" | 50 | 334 | 3 | - | - | 95.579 † | - | - |
| **Some animals show atypical behavior with poses very different to healthy ones (low humidity)** | | | | | | | | | |
| 100 fruit flies (2) | 10'00" | 34 | 538 | 3 | 217 | 99.905 | **99.277** | 0.546 | 0.177 |

**Supplementary Table 7: Results of manual validation for videos that do not fulfil some of the general video conditions listed in Supplementary Material B.1.** The column headers are defined in Supplementary Table 5. The first two videos were recorded in a compressed lossy video formats, .avi (FMP4 compression code) and .MOV (avc1), respectively. Lossy video formats may deleted pieces of information that could be important to identify the animals. The video of 10 fruit flies was recorded in a retroilluminated circular arena with conic walls. During the video, at regular time intervals a LED turns on and off. In images from retroilluminated arenas animals appear mainly as dark blobs, and the visual features of the body of the animals are less visible resulting in a more difficult identification. In addition, changes in light conditions can be problematic, since the appearance of the animals can change. This video was recorded by Clara Ferreira. The video of 14 ants has multiple shadows of animals which make the segmentation difficult. Furthermore, it is not illuminated with indirect light creating reflections in the body of the animals. This video was given by Andrew. I. Bruce and Nico Blüthgen (*Diacamma indicum* ants, preliminary tracking tests). The animals in the video of 60 flies have very low locomotor activity levels, and the first video of 100 female flies contains dead animals due to the after effects of the anaesthesia with $CO_2$. Images of animals during immobility periods represent a small set of the poses that animals can show over the duration of the video. A network trained with images of immobile animals is likely to fail to identify an animal when it starts moving or changes its pose. Moreover, the video of 60 flies has poor contrast and was segmented with a bad selection of preprocessing parameters. In the second video of 100 flies some animals show atypical behaviours like rolling on their back. In particular, there is an animal rolling on its back in the first global fragment. After with this global fragment, other animals which roll on their backs are identified as this animal.

**Table legend:** †: Estimated accuracy (see section B.2.7).

| Video | Pixels per animal (segmentation) | Pixels per animal (identification) | File size (Gb) | Tracking time | Protocol | Accuracy |
|---|---|---|---|---|---|---|
| 8 zebrafish | 331 | 599 | 9.14 | 0:30:51 | 2 | 99.969 |
| | 168 | 361 | 5.14 | 0:19:38 | 2 | 99.893 |
| | 63 | 193 | 2.29 | 0:19:04 | 2 | 99.890 |
| | 23 | 100 | 1.12 | 0:29:34 | 2 | 99.618 |
| | 9 | 61 | 0.57 | 0:23:42 | 2 | 96.690 |
| | 2 | 28 | 0.20 | 4:15:31 | 3 | 48.366 |
| 60 zebrafish | 303 | 578 | 228.02 | 5:18:41 | 2 | 100 |
| | 163 | 367 | 128.26 | 5:14:47 | 2 | 99.998 |
| | 64 | 193 | 57.00 | 7:05:52 | 2 | 99.998 |
| | 28 | 104 | 27.92 | 15:18:29 | 3 | 99.989 |
| | 12 | 63 | 14.25 | 12:42:58 | 3 | 89.187 |
| | 3 | 28 | 5.13 | 1 day, 22:33:39 | 3 | 19.572 |

**Supplementary Table 8: Results of human validation for 6 different image resolutions.** Pixels per animal after segmentation and identification stages, file size, tracking time, Protocol that idtracker.ai used and human validated accuracy. We give results for 6 different resolutions and two videos (8 and 60 zebrafish)

.

| Video | $\sigma$ gaussian kernel | Tracking time | Protocol | Accuracy |
|---|---|---|---|---|
| 8 zebrafish | 0.0 | 0:30:51 | 2 | 99.969 |
| | 0.5 | 0:39:00 | 2 | 99.876 |
| | 1.0 | 0:23:59 | 2 | 99.902 |
| | 2.0 | 0:21:44 | 2 | 99.865 |
| | 3.0 | 0:34:09 | 2 | 99.684 |
| | 4.0 | 0:28:12 | 2 | 99.300 |
| | 5.0 | 2:31:30 | 3 | 81.215 |
| 60 zebrafish | 0.0 | 5:18:41 | 2 | 100 |
| | 0.5 | 8:48:47 | 2 | 99.997 |
| | 1.0 | 6:41:33 | 2 | 99.998 |
| | 2.0 | 11:23:53 | 2 | 99.996 |
| | 3.0 | 11:32:12 | 2 | 99.995 |
| | 4.0 | 1 day, 11:39:10 | 3 | 98.367 |
| | 5.0 | 22:22:50 | 3 | 85.359 |

**Supplementary Table 9: Results of human validation for 7 values of standard deviation of a Gaussian blurring**. Standard deviation of the Gaussian kernel used, tracking time, Protocol that idtracker.ai used and human validated accuracy. We give results for 7 different values of the standard deviation and two videos (8 and 60 zebrafish)

.

| Video | Compression | File size (Gb) | Tracking time | Protocol | Accuracy |
|---|---|---|---|---|---|
| 8 zebrafish | raw video | 9.14 | 0:30:51 | 2 | 99.969 |
| | H264 | 3.83 | 0:34:36 | 2 | 99.895 |
| | MPEG4 | 0.82 | 0:22:02 | 2 | 99.896 |
| 60 zebrafish | raw video | 228.02 | 5:18:41 | 2 | 100 |
| | H264 | 95.48 | 6:55:03 | 2 | 99.997 |
| | MPEG4 | 12.92 | 6:20:37 | 2 | 99.998 |

**Supplementary Table 10: Results of human validation for 2 video compression algorithms**. Raw video and H264 and MPEG-4 compressions, corresponding file size, tracking time, protocol that idtracker.ai used and human validated accuracy. We give results for a video of $8$ zebrafish and $60$ zebrafish

.

| Video | Transfer learning | Protocols cascade time | Protocol | Accuracy | Accuracy identified animals | % animals identified |
|---|---|---|---|---|---|---|
| 8 zebrafish | no | 0:24:48 | 2 | 99.969 | 99.997 | 99.972 |
| | yes | 0:07:52 | 2 | 99.904 | 99.984 | 99.920 |
| 60 zebrafish | no | 2:08:27 | 2 | 99.998 | 100 | 99.998 |
| | yes | 1:18:49 | 2 | 99.962 | 100 | 99.962 |
| 72 fruit flies | no | 2 days, 12:08:58 | 3 | 99.997 | 100 | 99.997 |
| | yes | 3:30:25 | 2 | 95.584 | 99.452 | 96.111 |

**Supplementary Table 11: Results of human validation for videos tracked using transfer learning**. Time spend during the training and identification protocols cascade, protocol used to track the video, accuracy, accuracy of the identified animals (note that some individual can be left unidentified during the tracking process) and percentage of animals identified. We give results for videos of 8 zebrafish, 60 juvenile zebrafish and 72 fruit flies. For the zebrafish videos, the transfer was done from an identification network of a tracked video of 100 zebrafish. For the 72 fruit flies, the transfer was performed from an identification network of a tracked video of 100 fruit flies. Only the last two fully-connected layers of the network were trained when performing transfer learning

| Process | Parameter | Required | Type | Range | Default |
|---|---|---|---|---|---|
| Preprocessing | Minimum intensity | × | int | $[0, 255]$ | 0 |
| | Maximum intensity | ✓ | int | $[0, 255]$ | 135 |
| | Number of animals | ✓ | int | – | 0 |
| | Maximum area | × | int | $[0, 60000]$(px) | 60000 |
| | Minimum area | ✓ | int | $[0, 10000]$)(px) | 150 |
| | Region of interest(s) | × | bool | – | False |
| | Background subtraction | × | bool | – | False |
| | Check segmentation | × | bool | – | False |
| Tracking / identification network | CNN model | × | int | $[0, 11]$ | 0 |
| | Learning rate | × | float | $\mathbb{R}^+$ | 0.005 |
| | Dropout | × | float | $[0, 1]$ | 1.0 |
| | Optimizer | × | str | SGD or Adam | SGD |
| | Trainable layers | × | str | all or fully | all |
| | Save folder | × | str | – | Automatically generated |
| | Knoledge transfer | × | str | – | - |

**Supplementary Table 12: idtracker.ai parameters.** List of necessary and optional parameters available in the idtracker.ai GUI.

# Appendix B

# Supplementary Notes

## B.1 General video conditions

It is advisable to adhere to some guidelines during the realisation of videos of freely-moving animals. Here follows a list of conditions that allow to maximise the probability of success and the accuracy of the tracking.

- **Resolution**. The higher the number of pixels per individual, the more information to distinguish it from the rest. Notice that, on the downside, the additional information makes the algorithm less time-efficient. Check Supplementary Tables 5 and 6 for the average number of pixels per animal in each of the videos tracked.
- **Frame rate**. The frame rate must be high enough for the blobs associated with the same individual to overlap in consecutive frames, when moving at average speed. A low frame rate—with respect to the average speed of the animals—can cause a bad fragmentation of the video: An essential process in the tracking pipeline, that allows to collect images belonging to the same individual and organise them in fragments. On the contrary, excessively high frame rates will make the information coming from the analysis of the fragments highly redundant. This will increase the computational time necessary to track the video, without guaranteeing an improvement of the identification of the individuals. In the examples provided in this paper, the frame rate ranges from $25$fps to $50$fps.
- **Duration**. The length of the video for which the system works depends on the number of animals, the distribution of images per individual fragment and the number of pixels per animal. For few animals (8 zebrafish) we can track videos as short as $\approx 18$ sec ($\approx 500$ frames at 28 fps. For large groups we can track videos as short as 1 min ($\approx 1950$ frames at 32 fps). The system works for longer videos as far as the overall conditions do not change abruptly in different parts of the video. Very large videos with many animals will require a high amount of RAM and could block your computer.
- **Video format**. The system works with any video format compatible with OpenCV. We recommend uncompressed or lossless video formats: Some compression algorithms work by deleting pieces of information that could be crucial for the identification of the individuals. However, we have successfully tracked videos with compressed formats: .avi (FPM4 video codec) and .MOV (avc1 video codec) (see Supplementary Table 7).
- **Illumination**. Illumination has to be as uniform as possible, so that the appearance of the animals is homogeneous along the video. We recommend using indirect light either by making the light reflect on the walls of the setup, or by covering the setup with a light diffuser as shown in Supplementary Figures 3 and 4. Although, we have also tracked videos with retroilluminated

arenas (see Supplementary Table 7), recall that the tracking systems relies on visual features of the animals that this type of illumination could hide.

- **Definition and focus**. Images of individuals should be as sharp and focused as possible for their features to be clearly displayed along the entire video. When using wide apertures on the camera, the depth of field can be quite narrow. Make sure that the plane of the sensor of the camera is parallel to the plane of the arena so that animals are focused in all parts of it. In addition, exposition time (shutter speed) should be high enough so that animals do not appear blurred when moving at average speed. Blurred and out of focus images are more difficult to be identified correctly.

- **Background**. The background should be as uniform as possible. To facilitate the detection of the animals during the segmentation process (see section B.2.1), the background colour has to be chosen in order to maximise the contrast with the animals. Small background inhomogeneity or noise are acceptable and can be removed by the user during the selection of the preprocessing parameters:

  - Static or moving objects much smaller or much larger than the animals can be removed by setting the appropriate maximum and minimum pixels size thresholds.
  - Static objects of the same size and intensity of the animals can be removed by selecting the option "subtract background" in the preprocessing tab.
  - Regions of the frame can be also excluded by selecting a region of interest.

- **Shadows**. Shadows projected by the individuals on the background can lead to a bad segmentation and hence, to a bad identification. Shadows can be diffused by using a transparent base separated from an opaque background (see Supplementary Figure 3) or by using a retroilluminated arena.

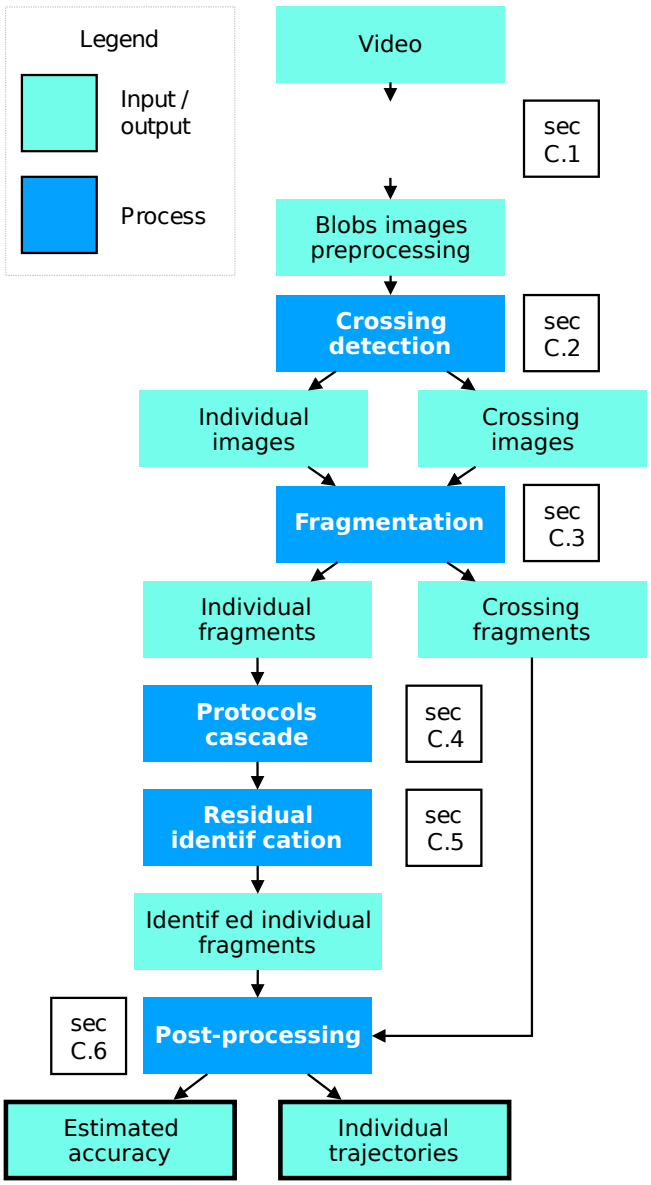
- **Reflections**. Reflections of individuals on the walls of the arena should be avoided: They could be mistaken for an actual individual during the segmentation process. Reflections in opaque walls can be reduced by using either very diffused light or matte walls. For aquatic arenas with transparent walls, reflections can be softened by having water at both sides of the walls. Furthermore, reflections can be removed by selecting an appropriate ROI.

- **Variability in number of pixels per animal**. The number of pixels in a blob is one of the criteria used to distinguish individual fish from crossings. An optimal video should fulfil the two following conditions. First, the number of pixels associated with each individual should vary as little as possible along the video. Second, the size an individual should vary as little as possible depending on its position in the arena. In any case, strategies to avoid misidentification are put in place, even in case of variable animal sizes (see section B.2.2).

## B.2 Algorithm

First, we introduce the work-flow of the algorithm. Subsequent sections will give further details on each of the components in the work-flow. The algorithm is composed of six computational cores highlighted in blue in Supplementary Figure SN.1. First, during the segmentation process the images representing either single or multiple touching animals are extracted from the video. In the remainder, we will refer to images representing a single individual as *individual images* and to images in which two or more individuals are touching as *crossing images*.

A model of the average area of the individuals, and later a convolutional neural network (CNN)—named crossing detector in the remainder—are used to discriminate between individual and crossing images.




**Supplementary Figure SN. 1:** Simplified algorithmic flow. Refer to the section specified next to each process block for details.

Each image extracted from the video is now labelled as either a single individual or a crossing. By means of an extra-safe protocol, we define collections of images in subsequent frames of the video in which the same individual (or crossing) is represented. We name these collections individual and crossing *fragments*, respectively.

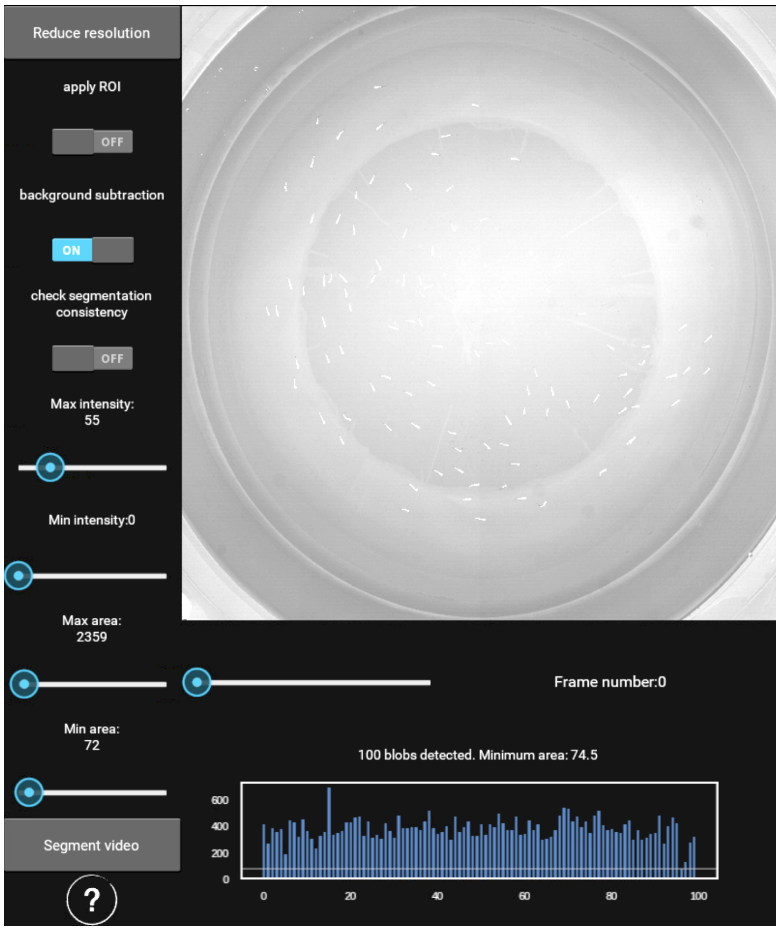
The fourth computational core is the gist of the algorithm. A subset of the collection of individual fragments, in which all the individuals are visible in the same part of the video is used to generate a dataset of individual images labelled with the corresponding identities. This dataset is then utilised to train a second CNN to classify images according to their identity. A cascade of increasingly encompassing training/identification protocols is put in place, so that an appropriate identification strategy is automatically defined by the algorithm, according to the complexity of the video. The idea underlying this family of methods is that the information gained from the first dataset of labelled images will allow either to accurately assign the entire collection of individual fragments, or to increase the first dataset by incorporating safely identified individual fragments throughout the video.

The knowledge acquired during the protocol cascade is used to identify the individual fragments that were not used to train the identification CNN. In the remainder, we will refer to this operation

as *residual identification*.

Finally, trivial identification errors are corrected by a series of post-processing routines, and the identity of the crossing fragments is inferred in a last computational core.

### B.2.1 Segmentation




**Supplementary Figure SN. 2:** Grayscale image thresholding for blob segmentation. idtrackerai has been tested with average individual blob areas of $\sim$ 300pxs. The resolution reduction button (top-left) allows to introduce a downsampling factor, to be applied to the entire frame, and consequently to the segmented blobs. The tree switches on the top allow to consider a predefined ROI, compute and subtract a model of the background. It is possible to activate a control on the number of blobs detected in each frame: If the segmentation returns more blobs than animals to be tracked in a frame or a collection of frames, the user will be asked to specify new segmentation parameters to be applied only in those frames. Finally, the user can define ranges of both acceptable intensities and blob areas, by adjusting the *Maximum* and *Minimum* intensity and area thresholds respectively.

idtracker.ai tracks the individuals by relying on their visual features. Hence, given a frame of the video, it is necessary to distinguish between pixels associated to individuals and background. According to the standard notation adopted in computer vision, we refer to a collection of connected pixels which is not part of the background as a *blob*.

The segmentation process has four main steps. First, the user can define a region of interest to be applied on each frame of the video. In this way it is possible to exclude, for instance, walls which may contain reflections of the animals.

Second, each frame is normalised with respect to its average intensity to correct for illumination fluctuations. It is also possible to perform background subtraction by generating a model of the background calculated as the average of a collection of frames obtained via subsampling the video.

Then, blobs of pixels corresponding to animals are detected by intensity thresholding and subsequent labelling of connected components. The intensity thresholds that allow to distinguish the individuals from the background are specified by the user. Often, intensity is not enough to segment the animals in the entire video. For this reason, it is also possible to specify a minimum and a maximum area (number of pixels) for a blob to be acceptable. For instance, these parameters allow to exclude dust during the segmentation.

All these operations are carried out in an intuitive way by using the idtracker.ai graphic user interface, where both the intensity and area thresholds can be adjusted by observing their effect in real time on the frame, see Supplementary Figure SN.2.

The software currently supports only grayscale video segmentation. Frames captured from a color video will be automatically mapped to grayscale.

*Remark* 1 (On background subtraction). Background subtraction is often useful when trying to segment a video in which a static object has the same intensity level as the individuals one wants to segment (see Supplementary Material B.1).

### B.2.2 Detection of individual and crossing images

The training/identification process allows to identify only images representing single individuals. Thus, a crucial point of the algorithm is the discrimination of individual and crossing images. In order to differentiate between these two classes, we apply a series of three different algorithms on the images segmented from the video.

First, we use two heuristics to detect images that in all likelihood correspond to a single animal (sure individual image) and crossing animals (sure crossing images), respectively. Then, we use these sure individual images and sure crossing images to train a neural network. Finally, the trained network is used to label ambiguous (not sure) images as either crossing or individual images.

#### Model area

We build a model of the area of the individuals by taking into account portions of the video in which the number of segmented blobs corresponds to the number of animals declared by the user. In case there is no frame in which this condition is fulfilled, the tracking cannot proceed and an error is raised. Let $\mathcal{C} = \{b_1, \ldots, b_n\}$ be the collection of the blobs segmented from these parts of the video and $A = \{area\,(b_i) \text{ for every } b_i \in \mathcal{C}\}$ the collections of the corresponding individual areas, where the function $area(b_i)$ counts the number of pixels corresponding to the blob $b_i$. The model area is defined by $m_A = \text{median}\,(A)$ and the standard deviation $s_A = \sigma\,(A)$. Let $b$ be a blob, we define

$$\gamma\,(b) = \begin{cases} \text{is an individual} & \text{if } |area\,(b) - m_A| < 4 \cdot s_A \\ \text{is a crossing} & \text{otherwise} \end{cases} \tag{B.2.1}$$

A model based exclusively on the area of the blobs can easily fail when the individuals' body is not rigid (e.g. fish or mice), can suddenly change shape (e.g. a fly with opened or closed wings), or under heterogeneous lighting conditions. Even more complex situations can arise when animals can move freely in 3 dimensions (e. g. fish swimming at different depths). In this latter case, one individual can be almost completely occluded by a second one, causing the model area to fail.

#### Blobs overlapping in subsequent frames

The second heuristic is based on the overlapping of blobs in subsequent frames: This allows to select sure crossing and individual images depending on the merging or splitting of consecutive, overlapping blobs. We recall that a blob is a collection of connected acceptable pixels in a certain frame, where a pixel is considered acceptable depending on its intensity value and the thresholding

described in section B.2.1. Let $b_1$ and $b_2$ be two blobs. We say that the two blobs overlap if and only if $b_1 \cap b_2 \neq \emptyset$, where the intersection $b_1 \cap b_2$ is the intersection between sets of pixels. See Supplementary Figure SN.3 for an example.

Let $B_i = \{b_{i,1}, \ldots, b_{i,n}\}$ be the collection of blobs segmented from the $i$th frame of a video $\mathcal{V}$. For every blob $b_{i,j} \in B_i$ we derive the collections of blobs overlapping with $b_{i,j}$ in frames $(i-1)$ and $(i+1)$. We call these collections the sets of previous and next blobs of $b_{i,j}$, denoted by $P_{b_{i,j}}$ and $N_{b_{i,j}}$, respectively.

Let $b$ be a blob. We say that $b$ is a blob associated with a *sure individual image* if:

a) $b$ is an individual according to eq. (B.2.1);

b) $|P_b| = |N_b| = 1$, i. e. the blob is overlapping with one and only one blob both in the previous and subsequent frame. The notation $|\cdot|$ indicates the cardinality of a set, i. e. the number of elements of the set.

c) for every $b_p$ and $b_n$ in the past and future overlapping history of $b$ $|P_{b_p}| \leqslant 1$ and $|N_{b_n}| \leqslant 1$.

Symmetrically, we say that $b$ is associated with a *sure crossing image* if:

a) $b$ is a crossing according to eq. (B.2.1);

b) $|P_b| > 1$ or $|N_b| > 1$.

or

a) $b$ does not satisfy the model of the area;

b) $|P_b| = |N_b| = 1$;

c) for some $b_p$ and some $b_n$ in the past and future overlapping history of $b$ $|P_{b_p}| > 1$ and $|N_{b_n}| > 1$.

### Crossing detector

The methods described in sections B.2.2 and B.2.2 can be used on any video in order to generate a dataset $\mathcal{D}_{ic}$ of *sure individual* and *sure crossing* images. With this dataset we train a CNN in the task of distinguishing crossing and individual images. We call this model crossing detector (CD). In the following paragraphs we will describe the preprocessing, architecture, hyperparameters and stopping criteria used to define and train this particular model.

**Preprocessing.** Let $b$ be a blob segmented from a video $\mathcal{V}$ and $I_b$ the image generated by cropping a rectangular bounding box around the centroid of $b$, such that all the pixels of $b$ are represented in $I_b$. We first consider a dilation $b^\star$ of $b$ generated with a $5 \times 5$ kernel. We assign value $0$ to every pixel in $I_b$ which is not in $b^\star$. In order to overcome the sensitivity of CNNs with respect to rotation, we compute the first principal component of the cloud of pixels defined by $b$ and then rotate and crop $I_b$ such that the first principal component forms and angle of $\frac{\pi}{4}$ with the x axis. After the rotation, the size of each image is set to be the maximum of the largest side for all the bounding boxes among the collection of sure crossing images. Then, the images are resized to $40 \times 40$ pixels. The resizing improves both the time and memory efficiency of the algorithm. Finally, each image $I \in \mathcal{D}_{ic}$ is standardised as $I_s = \frac{I - \mu(I)}{\sigma(I)}$ (see sample images in Fig 1, Panel d in the main text).

**Architecture.** See Supplementary Table 1 (crossing detector). Both convolutional and fully-connected layers are initialised using Xavier initialisation [14]. Biases are initialised to $0$.

**Loss function.** Let $(x, l_i)$ be a labelled image, where $l_i$ is the label in one-hot encoding, i. e. $l_0 = [1, 0]$ is the label associated with $x$ if $x$ is a crossing image, and $l_1 = [0, 1]$ if $x$ is an individual. We compute the loss function associated to $(x, l_i)$ as a weighted cross-entropy:

$$\mathcal{L}(x, l_i) = -w_i \sum_{j=0}^{n-1} l_i(j) \log s(a_j) = -w_i l_i(i) \log s(a_i), \tag{B.2.2}$$

where $s(a_i) = \frac{e^{a_i}}{\sum_{j=0}^{n-1} e^{a_j}}$ is the softmax function applied to the activation $a_i$ of the $i$th unit of the last layer of the network, with $j$ varying among all classes, in this case $j \in \{0, 1\}$; $w_i$ is the weight associated with $l_i$ and computed as

$$w_i = 1 - \frac{|L_i|}{\sum_{j=0}^{n-1} |L_j|},$$

where $|L_i|$ is the number of training samples belonging to the class $l_i$ and $j$ varies in the set of all the classes of the dataset (only two in this case: individual and crossing). The weighting allows to deal with the potentially unbalanced dataset $\mathcal{D}_{ic}$. Indeed, we prefer to collect all the possible examples of sure-crossing and sure-individual images available in a given video, rather than force $\mathcal{D}_{ic}$ to be balanced in the number of samples per class. After dividing the dataset $\mathcal{D}_{ic}$ in batches of $X_i$ of $50$ images, we optimise by considering the mean $\mu(\mathcal{L}(X_i))$ using the algorithm described [15], with the hyperparameters suggested in the paper. The learning rate is set at the initial value of $0.005$.

*Remark* 2. (On the softmax function) In general, the softmax is equipped with an extra parameter called temperature. We omit discussing it in the formula, since we always set it to $1$.

**Training and validation set.** Before training, the dataset of *sure crossing* and *sure individual* images is split into two parts: $90\%$ of the images are used for training, i. e. the weights of the network are updated in order to minimise the error (loss function) with respect to the labels associated with this set of images. We call this portion of the dataset the training set, denoted by $T$. The remaining $10\%$ of images–the validation set $V$–are used to evaluate the generalisation power of the network. For this reason, the performances of the model on the validation set are used to stop its training. In section B.2.2, we shall discuss in more detail the algorithm used to stop the training of the network.

**Accuracy.** We measure the accuracy of the network by comparing the predictions generated by the softmax computed on the activation of the last layer of the network, with the labels associated with each image in both the training and the validation set. Hence, let $|V|$ be the number of images in the validation set, $P_V = \{p_1, \ldots, p_n\}$ the ordered predictions generated by a forward pass of these images in the network, and $L_V = \{l_1, \ldots, l_n\}$ the corresponding labels. Let $A_V$ be the set of *correct predictions*, defined as $A_V = \{p_i \text{ s.t. } p_i = l_i \text{ for } p_i \in P_V,\ l_i \in L_V\}$. We define the overall accuracy of the network as

$$Acc_V = \frac{|A_V|}{|V|}. \tag{B.2.3}$$

We will also take into account the accuracy on each of the inferred classes. Let $c^\star$ be a class (in this case $c^\star$ could be either the crossing or the individual class). The set

$$A_V(c^\star) = \{p_i \text{ s.t. } p_i = l_i = c^\star \text{ for } p_i \in P_V,\ l_i \in L_V\},$$

corresponds to the predictions equal to their associated labels and attributed to the particular class $c^\star$. In this case the class-accuracy is defined as

$$Acc_V(c^\star) = \frac{|A_V(c^\star)|}{|\{l \in L_V \text{ s.t. } l = c^\star\}|}. \tag{B.2.4}$$

Symmetrically, we define the error and the class-error as $1 - Acc_V$ and $1 - Acc_V(c^\star)$, respectively.

**Training stopping criteria.** While training the network, we verify the goodness of its outputs by computing both the loss function and the accuracy on the validation set $V$ (see sections B.2.2 and B.2.2). This procedure gives a reasonable control on the actual classification power of the network on new unlabelled images. Thus, it is crucial to stop the training of the network to prevent two main behaviours. On the one hand, we want to prevent overfitting: A too exact representation of the trainig data, that will prevent the network from generalising on new data points. On the other hand, it is desirable to stop the training in case the error cannot be further minimised, i. e. the loss function reached a plateau.

More formally, we call an epoch a complete training pass on the set $T$, concluded with the evaluation (of both loss and accuracy functions) on the validation set $V$. Let $\mathcal{L}_i(T)$ and $\mathcal{L}_i(V)$ be the value of the loss function after the epoch $i$. We define

$$d(i^\star) = \mathcal{L}_{i^\star}(V) - \mu\left(\{\mathcal{L}_j(V)\}_{i^\star-10\leqslant j<i^\star}\right) \tag{B.2.5}$$

as the difference between the loss value in validation at $i^\star$, and the mean of the loss values of the previous $10$ epochs. We stop the training at epoch $i^\star > 10$ if one of the following conditions holds.

a) The network is overfitting: $d_i > 0$ for every $i^\star - 5 < i \leqslant i^\star$;

b) the loss reached a plateau: $|d_{i^\star}| < 0.05 \cdot 10^{\log_{10}(\mathcal{L}_{i^\star}(V))-1}$;

c) the network reached class-accuracy $1$ on all the classes, for every sample in the validation set. See eq. (B.2.3);

d) the loss (error) is zero: $\mathcal{L}_{i^\star}(V) = 0$.

**Crossing detection.** Let $\Delta$ be the set of parameters learnt by training the CD as described above. Let us denote the trained model as $\mathrm{CD}(\Delta)$. We create the test set $\mathbb{T}$ of unlabelled images by considering all the images that are not either *sure individuals* or *sure crossings*. The trained model acts as a function taking as an input an image $I \in \mathbb{T}$ and outputting a prediction as the softmax computed on the activation of the last layer of $\mathrm{CD}(\Delta)$. We recall that the softmax is the function defined as

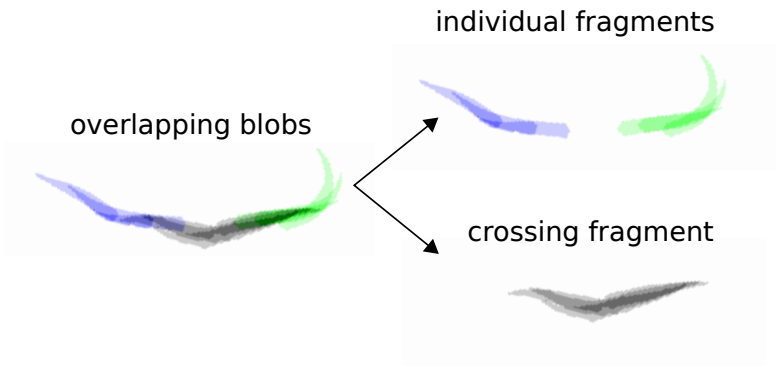
$$s(a_i) = \frac{e^{a_i}}{\sum_j e^{a_j}}, \tag{B.2.6}$$

where $a_i$ is the activation of the $i$th unit of the last layer. Since the CD classifies images in two classes, its last layer is composed by two units. Hence, given an image $I$, we obtain

$$\mathrm{CD}(\Delta)(I) = \{\, s_1, s_2 \,\},$$

where for brevity we set $s_i = s(a_i)$. If $s_1 > s_2$ the image is classified as a crossing, and as an individual otherwise.

**Exceptions** It is possible that during training the loss value diverges to infinity. In this case a warning is produced, and the algorithm falls back into a crossing-individual images discrimination process based only on model of the area of individual blobs (see section B.2.2). In case the criteria forcing the training to stop are never reached, we set a maximum number of $100$ epochs for the training of the CD. If this threshold is reached, a warning is produced and the training is stopped. The parameters computed in the last iteration will be used to classify individual and crossing images.




**Supplementary Figure SN. 3:** An individual fragment built by considering the blobs' overlapping in subsequent frames.

### B.2.3 Fragmentation

At this stage of the algorithm, the images segmented from the video (see section B.2.1) are labelled either as individual or crossing, following the protocols described in section B.2.2. A very careful dynamical analysis of the segmented blobs allows to create collections of images associated with the same individual (or crossing) in subsequent frames. In the remainder, we will refer to these collections as *individual* and *crossing fragments*. See Supplementary Figure SN.3 for an example of fragments and its decomposition in individual and crossing components.

The method used to create these types of fragments is based on the one hand on the classification of the images into crossing and individual categories; on the other hand, it considers the overlapping of the blobs associated with these images. We start by introducing some notations. Then, we will describe the algorithm to generate individual and crossing fragments in two separated sections.

Let $\mathcal{B} = \{b_{1,1}, b_{1,2}, \dots b_{1,n_1}, \dots b_{m,1}, b_{m,2}, \dots, b_{m,n_k}\}$ be the collection of segmented blobs, where the first index of the elements of $\mathcal{B}$ corresponds to the frame number, and the second to the order in which the blobs have been segmented. We recall, that given two blobs $b_1$ and $b_2$, we say that they overlap if and only if the intersection of their sets of pixels is not empty. Following the notation introduced in section B.2.2, given a blob $b_{i,j} \in \mathcal{B}$ we call the collections of blobs overlapping with $b_{i,j}$ in frame $(i-1)$ and $(i+1)$, $P_{b_{i,j}}$ and $N_{b_{i,j}}$ respectively.

#### Individual fragments

We iterate over the elements of $\mathcal{B} = \{b_{i,j}\}$ proceeding by frame number $i$ and then following the natural ordering induced by the second index. Let $b_{i,j}$ be a blob associated with the image $I_{b_{i,j}}$ labelled as an individual. We create individual fragments by considering only the future overlapping history of $b_{i,j}$. If $b_{i,j}$ is not yet part of any individual fragment, we associate with $b_{i,j}$ a unique fragment identifier $\alpha$ (i. e. $b_{i,j}$ is the blob intiating an individual fragment). To simplify the notation let $b_{i,j} = b$, and $N_{b_{i,j}} = N$. We consider two cases:

case 1: $|N| > 1$. The blob $b$ in frame $i$ overlaps with more than one blob in frame $i+1$, hence it is the only blob (and image) associated with the individual fragment $\alpha$.

case 2: $|N| = 1$. Let $n_b$ be the unique element of $N$. The fact that $b$ overlaps with a single blob in its future history is a necessary condition for $n_b$ to be part of the same fragment as $b$, but not sufficient. It could be that $|P_{n_b}| > 1$, thus we say that $n_b$ is in the same individual fragment as $b$ if and only if $|P_{n_b}| = 1$. We also require the image $I_{n_b}$ to be labelled as an individual.

If case 1 is verified we generate a new fragment identifier and continue iterating on the elements of $\mathcal{B}$. Otherwise, we apply the same algorithm to $n_b$ in order to enlarge the individual fragment $\alpha$ as much as possible. We stop adding blobs to the fragment whenever, during the iteration, a new candidate blob $n_{b_{last}}$ fulfils the condition in case 1. See algorithm 1 for the pseudocode.

**Algorithm 1** Assign individual fragment identifier to blobs

**procedure** ASSIGN_FRAGMENT_IDENTIFIER($\mathcal{B}$)
  $\alpha = 0$
  **for** $b \in \mathcal{B}$ **do**
    $cur_b = b$
    **if** $cur_b$ has no fragment identifier and $I_{cur_b}$ is an individual image **then**
      compute $N_{cur_b}$
      **if** $|N_{cur_b}| = 1$ **then**
        compute $P_{n_b}$ for $n_b \in N_b$
        **if** $|P_{n_{cur_b}}| = 1$ and $I_{n_{cur_b}}$ is an individual image **then**
          $n_{cur_b}$ is part of the fragment $\alpha$
          $cur_b = n_b$
          **while** $|N_{cur_b}| = 1$, $|P_{n_{cur_b}}| = 1$ and $I_{n_{cur_b}}$ is an individual image **do**
            $n_b$ is assigned to the fragment $\alpha$
            $cur_b = n_b$
          **end while**
        **else**
          $\alpha = \alpha + 1$
        **end if**
      **else**
        $\alpha = \alpha + 1$
      **end if**
    **else**
      continue
    **end if**
  **end for**
**end procedure**

### Crossing fragments

In the same setting of the previous section let $b_{i,j} = b$ be a blob associated with a crossing image. If $b$ is not yet equipped with a crossing fragment identifier, we generate a new identifier $\beta$. The conditions are almost equivalent to the individual fragments' case:

case 1: $|N| > 1$. The blob $b$ in frame $i$ overlaps with more than one blob in frame $i+1$, hence the crossing represented by $b$ is splitting. Thus $b$ is the only blob associated with the crossing fragment $\beta$.

case 2: $|N| = 1$ and $|P_{n_b}| = 1$ and $I_{n_b}$ is a crossing image. We add $n_b$ to the crossing fragment $\beta$.

In the second case, we try to extend the crossing fragment simply by iterating the algorithm on $n_b$, and verifying that both $P(n_b)$ and $N(n_b)$ have cardinality 1, and the unique element of $N(n_b)$ is associated with a crossing image.

The pseudocode presented in algorithm 1 can be easily adapted to work with crossing fragments, by modifying the *if*s and while conditions.

## B.2.4 Cascade of training/identification protocols

After fragmentation has finished, the training of the identification network begins. We would first like to give the reader some intuition regarding why it is possible to train an identification network in an automated manner. First imagine that we had at our disposal an all-knowing black box, that

looked at the set of fragments we have complied from one part of the video and then told us which fragment belonged to which individual. Remember that each fragment contains an entire set of images belonging to a single individual. Therefore, thanks to the information coming from the black box, we would effectively have at our disposal a set of labelled images and we could use standard supervised learning to train a classifier which can tell the individuals apart from one another. The trained network could then be used on other parts of the video to do identification.

In real life, we do not have access to such source of information, so we need to use some heuristics to generate our training dataset. In order to understand our heuristic, let's again remember that each fragment is supposed to contain images belonging to a single individual. We also know the total number of individuals in the video as this number is specified by the user. Consequently, if we can find one frame of the video, where the number of fragments which are present at that frame is equal to the total number of individuals, then we could be sure that each visible fragment in that frame belongs to a separate individual. We can then label the images within each fragment with the same label while every fragment of course has a different label. Next, we train our network on the resulting dataset of images and labels. This is the intuition behind how we achieve our aim without the help of an omniscient black box.

In order to train the identification network, we have designed three training protocols. The first protocol is the fastest and is able to deal with videos where animals are relatively well separated (crossings are not too frequent). The other two protocols handle more difficult scenarios, where crossings may be frequent, the setup lighting intensity may drift over time, the animals may change their features throughout the video (e. g. , posture, colour).

Each protocol relies on the information acquired and structures defined in the previous ones. In the following sections we will introduce some definitions and the main elements on which the fingerprint protocols are built. Then, we will discuss each of the three protocols from the simplest and fastest, to the most general and computationally expensive one.

### Global fragments

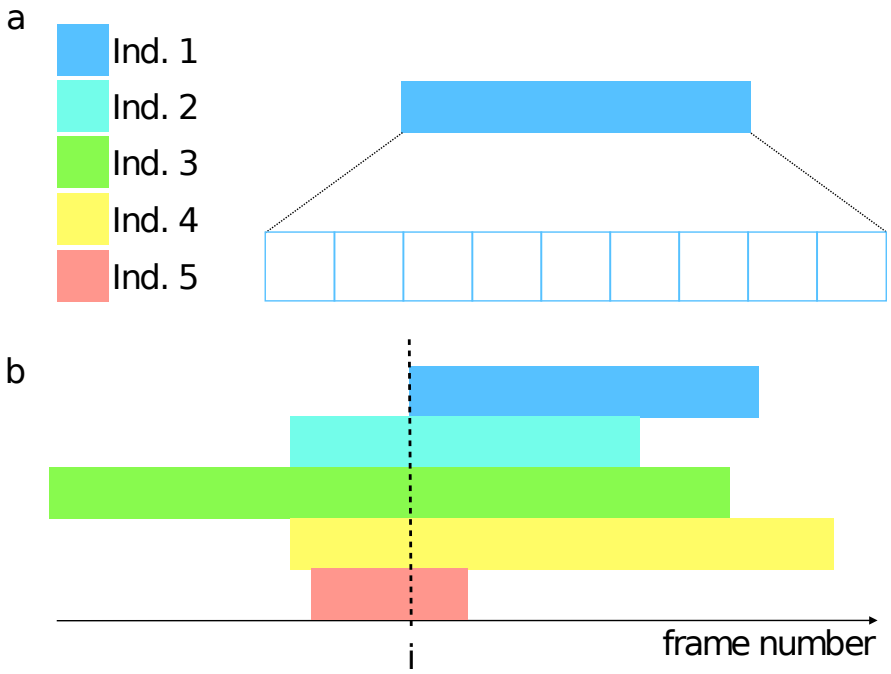
All the protocols rely on a strong, fundamental hypothesis: To learn the features characterizing each individual and consequently identify it, there must exist at least one portion of the video in which all animals are visible and separated.

Let $\mathcal{V}$ be a video in which the aforementioned condition is fulfilled in frame number $i$. We define a *global fragment* as the collection of individual fragments (see section B.2.3), whose images contain the ones extracted from the $i$th frame of the video and with the same number of fragments as the number of individual to be tracked. We call the minimum frame number in which this condition is satisfied the *core of the global fragment*. See Supplementary Figure SN.4 for a visual representation of a global fragment. We denote by $\mathcal{G}$ the set of all the global fragments in $\mathcal{V}$, whose shortest individual fragment counts at least $3$ images.

### Identification network

All the fingerprint protocols aim at finding different strategies in order to create datasets of images of the animals labelled with their identities. These datasets will be created from one, or a collection of global fragments and used to train the *identification CNN*, denoted in the remainder as idCNN. In the following paragraphs we define the architecture, the hyperparameters and algorithms used to train the idCNN.

**Preprocessing.** The images used to train and test the idCNN are preprocessed with an algorithm similar to the one described in section B.2.2. The images are obtained, aligned and standardised in the same way. The only difference is that the square images used to train the CD are resized to be square




**Supplementary Figure SN. 4:** Global fragment in a video of 5 individuals. A) Each individual in the video is associated with a certain label. An individual fragment is a collection of preprocessed images associated with the same individual. B) A global fragment is a collection of individual fragments that coexist at least in a frame of the video, the frame $i$ in the image.

images of size $40 \times 40$, while the training images of the idCNN are obtained as $\frac{\text{estimated body length}}{\sqrt{2}}$. The body length is estimated by considering the median of the diagonal of the images associated to each individual blob.

**Architecture.** See See Supplementary Table 1 (identification convolutional neural network). Both convolutional and fully-connected layers are initialised using Xavier initialisation [14]. Biases are initialised to 0.

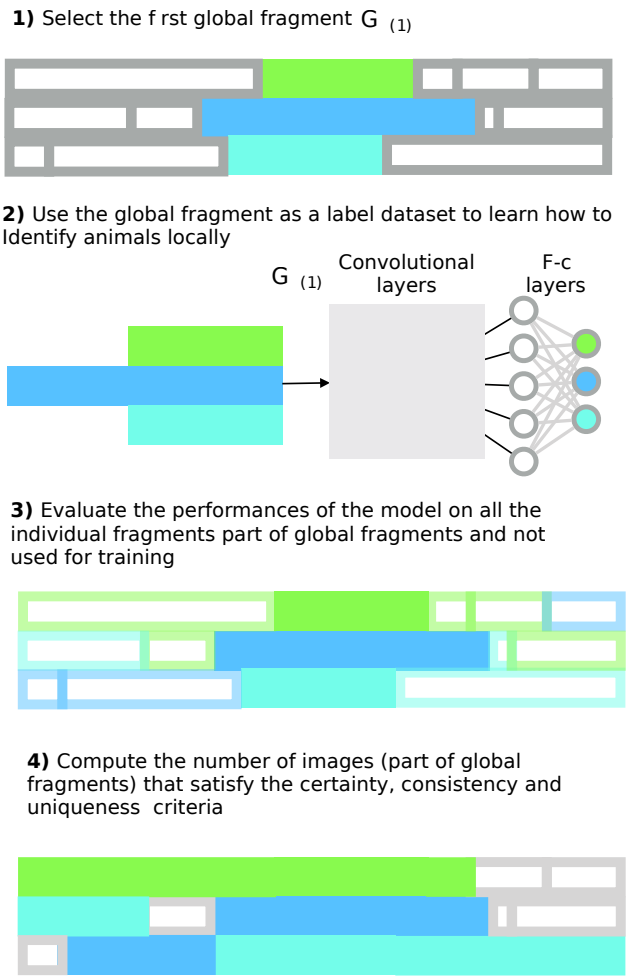
**Loss function.** The loss is a weighted cross-entropy eq. (B.2.2). The dataset given by a global fragment is potentially unbalanced: Every individual fragment $F_i \in G$ counts a certain number of images, say $n_i$. For every $F_i \in G$, we compute the weight $w_i$ of eq. (B.2.2) as

$$w_i = 1 - \frac{n_i}{\sum_j n_j},$$

where $j$ varies in $\{ F_1, \ldots, F_n \} \in G$. Thus, a larger loss is associated with individuals less represented in the dataset. We optimise using stochastic gradient descent, setting the learning rate to $0.005$.

**Training and validation set.** Every individual fragment $F_i \in G$ can be written as $F_i = \{(I_1, l_1), \ldots, (I_{n_i}, l_{n_i})\}$, where $(I_j, l_j)$ is a pair such that the label $l_j$ is the identity of the individual depicted in the image $I_j$. The dataset generated from $G$ is given by the union $\mathcal{D}_G = \cup_i F_i$. After a random permutation of the pairs $(I_j, l_j)$ in order to lose any temporal correlation between the images, we split $\mathcal{D}_G$ in the training and validation sets denoted by $T$ and $V$ respectively. These sets are composed of $90\%$ and $10\%$ of the available data, respectively.

**Accuracy.** The accuracy of the network is computed as the number of successfully classified images over the total number of images, according to eq. (B.2.3). We measure the single class accuracy




**Supplementary Figure SN. 5: Protocol 1.** The simplest fingerprinting protocol takes advantage of the information of a single global fragment (1) chosen according to the algorithm discussed in section B.2.4, in order to train the idCNN (2) (see section B.2.4). In step (3), the entire collection of individual fragments belonging to at least one global fragment is identified by using the classification provided by the model trained in (2) and according to the individual fragments identification algorithm described in section B.2.4. Finally, the performace of the model is evaluated (see section B.2.4).

following eq. (B.2.4). This second expression is fundamental when dealing with large groups, in order to evaluate the capability of the network to distinguish each of the individuals.

**Training stopping criteria.** See section B.2.2.

**Exceptions** It is possible that during training, the loss value diverges to infinity. In this case an error is raised and the algorithms stop its execution. Advanced users have the possibility to change the parameters of the idCNN (e. g. learning rate, dropout, layers' units count).

If the training of the idCNN is not stopped before 10000 epochs (passes over the entire dataset) a warning is produced and the training is stopped. The parameters computed in the last epoch will be used to continue the fingerprint protocol cascade.

### Protocol 1: One-global-fragment tracking

This protocol is based on the features learnt by considering the images belonging to a single global fragment. Thus, it is likely to be successful when the individual images are uniform along the entire video.

**Algorithm 2** Compute distance travelled in individual fragment

**procedure** COMPUTE_DISTANCE_TRAVELLED($F$)
  $c(F) = \{c_1, \ldots, c_n\}$ ▷ Centroids of the images
  distance_travelled $= 0$
  **for** $(c_i, c_{i+1})\ i \in \{1, \ldots, n-1\}$ **do** ▷ For each pair of subsequent blobs
    $d = \| c_{i+1} - c_i \|_2$ ▷ Compute the Euclidean distance
    distance_travelled $+= d$ ▷ Add it to the overall distance
  **end for**
  **return** distance_travelled
**end procedure**

**Choosing the global fragment.** Since the network will be trained on a single global fragment, its choice is fundamental. We aim at selecting the global fragment whose individual fragments are sets of images with high variability. This, in order to be as close as possible to the setting described in Supplementary Figure 2, where images are subsampled from the entire video, and hence uncorrelated in time.

We define the distance travelled in a global fragment $G$ as the minimum of the distance travelled in its individual fragments, as described in algorithm 2.

We choose the global fragment realising the maximum of the minimum distance travelled denoted as $G_{\sigma(1)}$. This procedure does not assure to get the global fragment whose images are maximally variable. However, there is a natural correlation between the distance travelled by an animal and the variability of the images stored in the corresponding fragment.

**Training.** After choosing $G_{\sigma(1)}$, we label the images belonging to each of its individual fragment with random, unique identities (think of increasing natural numbers). From this dataset of labelled images we create the training and the validation set, as described in section B.2.4. We train the idCNN as specified in section B.2.4. The training is interrupted automatically when one of the condition in section B.2.4 is satisfied. Let us call $\theta_0$ the set of parameters of the idCNN after training, and denote the trained model as idCNN$(\theta_0)\ (I)$.

**Single image identification.** We recall that a trained neural network acts as a function. We use the trained the idCNN in order to identify individual fragment not used for training. For every image $I$ in an individual fragment $F$, we can compute idCNN$(\theta_0)\ (I)$, obtaining $S_I = \{\ s_1, \ldots, s_n\ \}$, where

$$s_j = \frac{e^{a_j}}{\sum_i e^{a_i}}$$

is the softmax computed on the activity of the $j$th unit of the last fully connected layer. The identification network last layer has as many units as the number of individual to be tracked (see Supplementary Table 1, idCNN). By definition, $\sum_{s \in S_I} s = 1$. Thus, $S_I$ can be interpreted as the probability of the input image to represent each of the individuals. Each image is labelled with the identity realising the maximum of the softmax: $id\,(I) = \text{argmax}\,(S_I)$.

*Remark* 3. Given the relatively low number of parameters of idCNN ($\approx 200000$, see Supplementary Table 1, idCNN), and the usage of GPU computing, the single image identification step is time efficient, even when dealing with large groups of animals.

**Computing the identity probability mass function.** When considering an individual fragment, it is natural to take advantage of the hypothesis that all its images are associated with the same individual. We follow the assumptions and logic already presented in [1, Supporting text, Section

3.1]. Let $\Lambda_F = (f_1, \ldots, f_n)$ be the vector of identification frequencies associated with $F$, i. e. the vector whose components correspond to the number of images of $F$ assigned to the $i$th individual, computed as in algorithm 3. Under the assumption that all the images in $F$ are independent and that the probability to assign one image to the correct individual is twice as large as the probability to assign the image to any of the incorrect individuals, we compute for every identity $i \in \{ 1, \ldots, n \}$

$$P_1(F, i) = \frac{2^{\Lambda_F(i)}}{\sum_{i=j}^{n} 2^{\Lambda_F(j)}}, \tag{B.2.7}$$

where $\Lambda_F(i)$ is the $i$th component of the vector $\Lambda_F$.
The vector $P_1(F) = (P_1(F, 1), \ldots, P_1(F, n))$ is the probability mass function of $F$ being identified as one of the individuals.

**Model quality check and identification of individual fragments in global fragments.** While identifying the individual fragments belonging to the global fragments not used to train the idCNN($\theta_0$), we evaluate the goodness of the model. For every $G \in \mathcal{G} \setminus \{ G_{\sigma(1)} \}$ we proceed by verifying the following conditions, providing at the same time a temporary identification of the individual fragments in $G$:

1. $G$ **is certain:** A global fragment $G$ is certain if all the individual fragments $F_i$ in $G$ are also certain. A fragment $F$ is certain if $cert(F) \geqslant 0.1$. Let $a$ and $b$ be the indices (identities) realising the first and second maximum of $P_1(F)$ respectively, and $S_j$ the vector of softmax values of all the images assigned to the index $j$ in the fragment $F$. The function $cert(F)$ is defined as

$$cert(F) = \frac{\text{median}(S_a) \cdot P_1(F, a) - \text{median}(S_b) \cdot P_1(F, b)}{P_1(F, a) + P_1(F, b)}. \tag{B.2.8}$$

2. **Temporary identification of the individual fragments:** Let us consider the entire collection of individual fragments $\{F_i\}_{F_i \in G}$. We start by reordering it according to the maximum value of each $P_1(F_i)$. Let us denote the reordered collection of individual fragments as $\mathcal{F} = \{ F_{\rho(1)}, \ldots, F_{\rho(m)} \}$. We iterate on the individual fragments indexed as in $\mathcal{F}$. So, $F_{\rho(1)}$ is the individual fragment having maximum probability of being identified as the individual with identity $\iota = \text{argmax}\left(P_1\left(F_{\rho(1)}\right)\right)$. We set $\iota$ to be the temporary identity of $F_{\rho(1)}$ if two conditions are reached:

   1. There is no identified individual fragment coexisting with $F_{\rho(1)}$ with identity $\iota$.
   2. $\max_{f_j \in P_1\left(F_{\rho(1)}\right)} \left(P_1\left(F_{\rho(1)}\right)\right) > \frac{1}{\left|F_{\rho(1)}\right|}$, where $\left|F_{\rho(1)}\right|$ is the number of images in $F_{\rho(1)}$.

   We proceed to the next iteration by considering $F_{\rho(2)}$. If one of the aforementioned conditions is not satisfied, the individual fragment is marked as *non-consistent* as well as the entire global fragment $G$.

3. $G$ **is unique:** A global fragment $G$ is unique if the temporary identity of every $F_i$ in $G$ are unique within the global fragment.

We iterate on the global fragments in $\mathcal{G} \setminus \{ G_{\sigma(1)} \}$ by sorting them from the nearest to the farthest with respect to the distance in frames between their core and the core of $G_{\sigma(1)}$. See section B.2.4 for the definition of the core of a global fragment.

We now consider the number of images in all the individual fragments that are part of a global fragment. We will refer to this set of images in the remainder as the *global fragments' images* or the *images in global fragments*. If at least $99.95\%$ of these images are contained in global fragments considered acceptable with respect to the conditions listed in the previous paragraph, we interrupt the cascade of protocols and we pass to the residual identification described in section B.2.5. Otherwise, the second protocol is put in place.

**Algorithm 3** Compute identification frequencies in individual fragment

**procedure** GET_FREQUENCIES($F$) ▷ An individual fragment
  $S_F = []$ ▷ softmax array
  $ids = []$ ▷ identities array
  **for** $I \in F$ **do** ▷ For every image in $F$
    $s_I = \text{softmax}\left(\text{idCNN}\left(\theta_0\right)\left(I\right)\right)$ ▷ Compute softmax (GPU)
    $S_F.append(s_I)$
    $ids.append(\text{argmax}\left(s_I\right))$ ▷ Compute the identity (GPU)
  **end for**
  **return** frequencies = count(ids) ▷ Count assignment frequency
**end procedure**

### Protocol 2: Global-fragments-accumulation

The main aim of this protocol is to accumulate the images belonging to those global fragments, detected during protocol 1, that are simultaneously certain, consistent and unique. By iterating this procedure, it is possible to incorporate new images in the labelled dataset used to train the idCNN. This accumulation procedure allows to learn features able to grasp the individuals' variability through the video. See Supplementary Figure SN.6 for the flow of the algorithm of this protocol.

**Global accumulation.** Let $A_{-1} = \left\{ G_{\sigma(1)} \right\}$ be the set containing the first global fragment used for training, and $A_0 = \left\{ G_{1,1}, \ldots, G_{1,n} \right\} \subset \mathcal{G} \setminus G_{\sigma(1)}$ be the subset of global fragments that meet the conditions described in section B.2.4.

First, we fix the identities of the individual fragments belonging to the global fragments in $A_0$, since the images associated with these individual fragments will be used to train the idCNN.

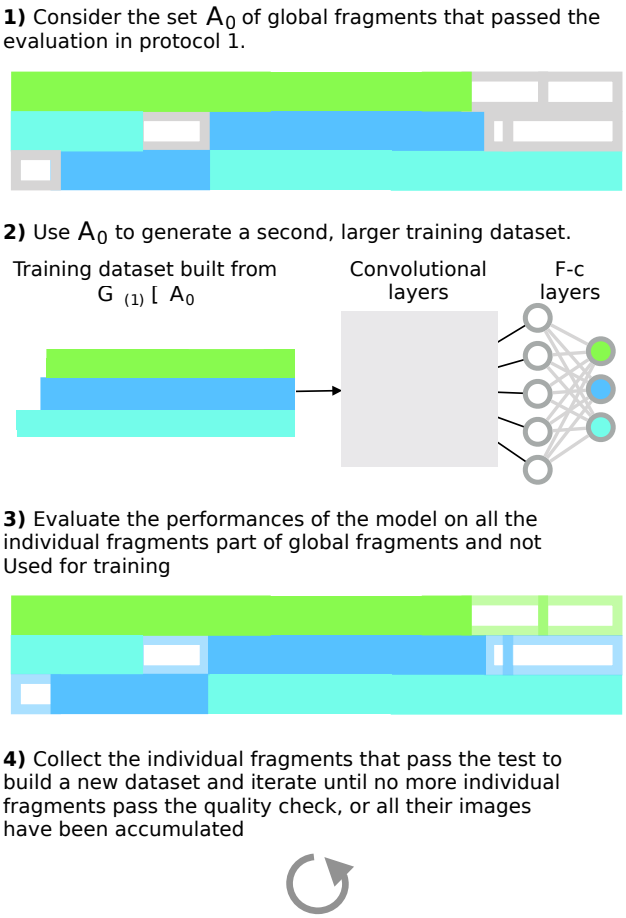
We build the dataset $\mathcal{D}_{A_0}$ by considering all the labelled images contained in every global fragments in $A_{-1} \cup A_0$. Note that, since an individual fragment can be shared by more global fragments, its images will be collected only once.

$\mathcal{D}_{A_0}$ is then split in the training and validation sets ($T_1$ and $V_1$), according to the proportions specified in section B.2.4 and an additional constraint: In the training and validation sets every individual can be represented by at most $3000$ images. If the amount of images associated with a certain individual in $\mathcal{D}_{A_0}$ exceeds this threshold, $3000$ images are randomly subsampled from this collection, by taking $1800$ samples from the images previously accumulated (images in $A_{-1}$), and the remaining $1200$ from the new set of accumulated images ($A_0$).

*Remark* 4. At every iteration of the accumulation, the permutation used to subsample the images representing the same individual changes in order to train the idCNN with maximally variable images.

We train the network using the stopping criteria listed in section B.2.2. Following the notation introduced in section B.2.4, we denote the idCNN model obtained after training as $\text{idCNN}\left(\theta_1\right)$. The accumulation process is iterated by defining $A_i$ as the set of acceptable global fragments in $\left\{\mathcal{G} \setminus G_{\sigma(1)}\right\} \setminus \cup_{j=0}^{i-1} A_j$. The set of new acceptable global fragments is computed by identifying the individual fragments not used for training and apllying the procedure described in section B.2.4.

**Partial accumulation.** Partial accumulation is a riskier accumulation strategy. It allows to include in the dataset of accumulated images single individual fragments, rather than entire global fragments. For this reason, before applying this strategy, we require that more than half of the images contained in the set of global fragments has been accumulated via global accumulation. Assume that this condition is reached at the iteration $\bar{i}$, then an individual fragment $F \in G$ for some global fragment $G \notin A_{\bar{i}}$ is accumulated if




**Supplementary Figure SN. 6: Protocol 2.** Flow of the $i$th iterative step of protocol 2. In this protocol fingerprints are built iteratively, by iterative accumulation. In (1) images belonging to individual fragments assigned with high certainty in the $i-1$th iteration are collected to form a new, broader dataset of labelled images (see sections B.2.4 and B.2.4). This dataset is used to fine-tune the idCNN in (2). Individual fragments in the video that still lack identity are identified (3). The most certain ones are used to initialise the next iteration step.

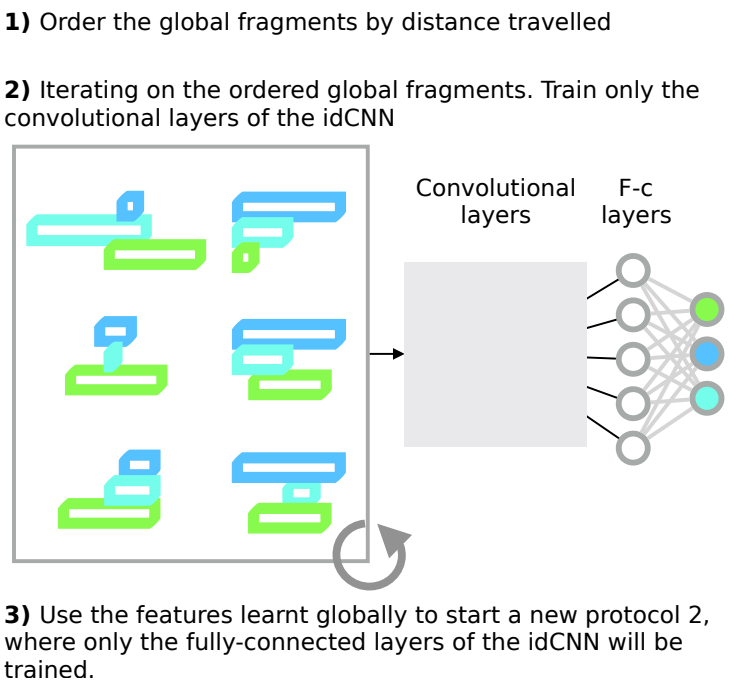
1. $cert\,(F) > 0.1$;
2. let $\gamma\,(F)$ be the set of individual fragments that coexist with $F$ in at least one frame of the video. $F$ can be accumulated if at least half of the elements of $\gamma\,(F)$ have already been accumulated;
3. the identity of $F$ is coherent with all the individual fragments in $\gamma\,(F)$, i. e. the assignment of a certain identity to $F$ does not create duplications.

If all these conditions are met, $F$ will be added to $A_{\bar{i}+1}$ as a single individual fragment.

**Accumulation stopping criteria.** We stop both the global and the partial accumulation processes if one of the two following conditions holds:

1. $99.95\%$ of the images in global fragments have been accumulated;
2. there are no more acceptable global or individual fragments.

**Evaluation of the accumulation.** If the number of images accumulated in the last iteration is less than $90\%$ of all the images in the global fragments, the accumulation is considered not acceptable and the third protocol is used. Otherwise we proceed to the identification of the individual fragments not identified during accumulation, see section B.2.5.




**Supplementary Figure SN. 7: Protocol 3.** The acquisition of the information that will be used to produce the fingerprints is split in two parts in this protocol. First, we consider the collection of global fragments, by deleting any previous identification (1). In step (2) we train the idCNN iterating on the set of global fragments. When iterating, we keep the weights of the convolutional part, as while the classifier is reinitialised. Finally we re-start protocol 2 by using the encompassing convolutional features learnt form the entire video, and training only the classification part of the network.

### Protocol 3: pretraining and accumulation

This last protocol allows to learn globally the features of the images representing the individuals by taking advantage of their local organisation in global fragments.

**Pretraining.** Given a video $\mathcal{V}$ let $\mathcal{G}$ be the set of global fragments as defined in section B.2.4. We rewrite the idCNN by considering it as the juxtaposition of its convolutional idCNN$_c$ and fully-connected idCNN$_f$ parts, with sets of parameters $\Gamma$ and $\Phi$, respectively. Here follows the list of processes involved in the pretraining algorithm:

(i) Consider the set $\sigma(\mathcal{G}) = \{ G_{\sigma(1)}, \dots, G_{\sigma(n)} \}$ of global fragments $\mathcal{G}$ ordered by distance travelled. See section B.2.4.

(ii) Iterate on the elements of $\sigma(\mathcal{G})$. Let $\mathcal{D}_{G_{\sigma(i)}}$ be the dataset of labelled images built at the $i$th iteration. Assign a random unique identity to each individual fragment: The aim is to learn features, and classify the individuals only locally. Generate both the training and validation sets as in section B.2.4.

(iii) Train the model using the parameters learnt during the previous iteration for the convolutional part idCNN$_c$ $(\Gamma_{\sigma(i-1)})$ and reinitialise idCNN$_f$. This step allows to learn convolutional filters optimised on the task of distinguishing the animals in $G_{\sigma(i-1)}$ based on their local labelling in the global fragment.

(iv) Conclude the training according to the conditions listed in section B.2.4.

(v) Iterate on $\sigma(\mathcal{G})$ until $95\%$ of the images stored in the global fragments have been used to train the network.

**Accumulation parachute.** After pretraining, we start the accumulation of reference images as in section B.2.4, but freezing the parameters of idCNN$_c$ learnt during pretraining along the entire accumulation. Thus, in the first step of this second accumulation we reinitialise only the fully-connected part of the idCNN. With these settings, we apply the accumulation protocol, by updating only the parameters $\Phi$, and starting by considering the global fragment $G_{\sigma(1)}$.

If more than $90\%$ of the images in the global fragments are accumulated during the accumulation, we proceed to the identification of the individual fragments not used for training section B.2.5. Otherwise, we will repeat the accumulation starting from $G_{\sigma(2)}$. If the accumulation fails even in this case, we repeat it with $G_{\sigma(3)}$ as a basis. Finally, we end the deep protocol cascade by selecting the accumulation in which the largest amount of images has been used for traninig and hence, already identified. By using the parameters of the idCNN learnt in the chosen accumulation, we proceed to the identification of the remaining individual fragments.

*Remark* 5. As pointed out in section B.2.4, the computation of the distance travelled cannot guarantee the images in $G_{\sigma(1)}$ to be maximally uncorrelated. Hence it is important, rather than assigning identities with a non-optimal model, to try and learn starting from different global fragments, that could incorporate images whose features are keys to maximise the amount of accumulated global fragments.

## B.2.5 Residual identification

After the fingerprint protocol cascade, it is necessary to identify those individual fragments that could not be accumulated, either because they are not included in any global fragment, or they gave a low certainty value during test. We recall that all the individual fragments already accumulated are endowed with both an identity and the $P_1$-vector. See section B.2.4 and eq. (B.2.7) for details about the identification of individual fragments during accumulation.

### Non-accumulated images identification

Let $\mathcal{U} = \{ F_1, \ldots, F_n \}$ be the set of individual fragments that are not identified during the protocols described in section B.2.4. First, we assign an identity to every image $I$ in every $F_i$ in $\mathcal{U}$. To do that, we pass every image through isCNN $\left(\Theta_{\text{final}}\right)$, where $\Theta_{\text{final}}$ are the parameters learnt during the fingerprint protocol cascade. Then $P_1\left(F_i\right)$ is computed for every $F_i \in \mathcal{U}$ according to section B.2.4 and eq. (B.2.7).

### Identification of non-accumulated individual fragments

When assigning the identity to an individual fragment, it is desirable to take advantage of the fact that the same identity cannot be assigned to two fragments that coexist in time. Given an individual fragment $F$, let $\overline{\gamma}\left(F\right) = \left\{ \overline{F_1}, \ldots, \overline{F_n} \right\}$ be the set of identified individual fragments coexisting with $F$ and not $F$ itself, and such that every $\overline{F_i} \in \gamma\left(F\right)$ is equipped with a $P_1$-vector. We integrate the information coming from the identified fragments coexisting with $F$ by following the approach of [1, Supporting text, Section 3.1]. We define the probability of the fragment $F$ to be assigned to the identity $i$ as

$$P_2\left(F, i\right) = \frac{P_1\left(F, i\right) \prod_{\overline{\gamma}(F)} \left(1 - P_1\left(\overline{F}, i\right)\right)}{\sum_{j=1}^{n} P_1\left(F, j\right) \prod_{\overline{F} \in \overline{\gamma}(F)} \left(1 - P_1\left(\overline{F}, j\right)\right)}. \tag{B.2.9}$$

where $n$ is the total number of animals.

Furthermore, we define the identification certainty as

$$\overline{cert}(F) = \frac{P_2\left(F, a\right)}{P_2\left(F, b\right)}, \tag{B.2.10}$$

where $a$ and $b$ are the indices (identities) that realise the first and second maximum of $P_1(F)$, respectively.

We compute $P_2(F) = (P_2(F,1), \ldots, P_2(F,n))$ and $\overline{cert}(F)$ for every individual fragment $F \in U$. We proceed to identify the fragments in $\mathcal{U}$ from higher to lower values of $\overline{cert}$. For every fragment we assign the identity $\iota = \text{argmax}_i(P_2(F,i))$. If there are two identities realising the maximum $P_2$, we do not identify the fragment (in the GUI this fragments are indicated with the identity 0 and black colour). If the fragment $F$ is identified, say with identity $i$, we set $P_1(F,i) = 1$ and $P_1(F,j) = 0, \forall j \neq i$ . Then, we recompute $P_2(F)$ and $\overline{cert}$ for every fragment in $\overline{\gamma}(F)$. According to eq. (B.2.9), all the fragments in $\overline{\gamma}(F)$ will have $P_2(F,i) = 0$. This prevents the assignment of the same identity to multiple coexisting individual fragment.

The process is iterated on $\mathcal{U} \setminus \{ F \}$, until all the fragments are either equipped with an identity or unsuitable for identification.

### B.2.6 Post-processing

The training/identification protocols and the residual identification assigned an identity to a as large as possible number of individual fragments. The methods involved in the post-processing stage of the algorithm take care of correcting trivial identification mistakes and, thereafter, to identify the individuals involved in crossings.

These processes are described in details in the following sections; here, we provide an intuition concerning the algorithms involved in both of them. It is possible to correct trivial identification errors by considering adjacent individual fragments assigned to the same individual. If the individual has to reach a supernatural speed in order to move from its position at the end of a fragment, to the position corresponding to the beginning of the next one, the identification is assumed to be incorrect. A series of heuristics allows to either assign a new (not necessarily different) identity to the fragments involved in the process, or renounce to their identification.

The idea underlying the identification of crossings is basically an informed interpolation of the individual trajectory. First, we consider a blob associated to a crossing. We workout the identities of each crossing individual by trying to split the blob by successive erosions. If the blob splits in smaller parts (say sub-blobs), we try to link each sub-blob to an already identified individual fragment. To do that, we consider two conditions. On one hand we evaluate the eventual overlapping of the sub-blobs we just obtained with identified, individual blobs segmented either in the next or the previous frame. In case the overlapping strategy fails, we seek individual blobs in adjacent frames with respect to the considered crossing, that can be linked to the sub-blob by using speed-constraints similar to the ones discussed in the previous paragraph.

#### Evaluate unrealistic identifications at fragment boundaries

Individual fragments are defined by considering the overlapping of blobs segmented from consecutive frames (see section B.2.3). Let us denote the frame numbers spanned by a fragment $F$ as $[f_s, f_e]$, where $f_s$ is the number of the frame from which the first blob associated to $F$ has been segmented. We say that two individual fragments $F_1$ and $F_2$ are *consecutive* if they share the same identity, say $i$, and $f_{1e} < f_{2s}$. We aim at evaluating the goodness of the identification of such fragments by comparing a model of the stereotypical speed of the individuals in the video, with the speed that the individual $i$ needs to reach to travel from its position in $f_{1e}$ to its new position in $f_{2s}$.

**Computation of the stereotypical speed:** The stereotypical speed is computed as follows by considering the speed of the animals in every individual fragment:

1. For every individual fragment $F$, let $(b_1, \ldots, b_n)$ be the blobs collected in $F$, and $(c_1, \ldots, c_n)$ their centroids.

2. We compute the speed of the animal in $F$ by considering the distance in pixels between subsequent centroids. Namely, $v_i = d\left(c_i, c_{i+1}\right)$ for $i \in \{\, 1, \ldots, n-1 \,\}$.

3. We define $v_{\max} = P_{99}\left(V\right)$, where $V$ collects the speeds computed from every individual fragment $F$.

**Evaluation of consecutive fragments:** We set to immutable the identity of all the fragments that have either been identified during the deep protocol cascade, or whose identity has been assigned during the residual identification with $\max_i(P_2(F)) \geqslant 0.9$.

Let $F_1$ and $F_2$ be the consecutive fragments described above. The speed at the boundary $s_{F_1 \to F_2} = \frac{d(c_{1e}, c_{2s})}{f_{2s} - f_{1e}}$ needed to connect the two fragment is *realistic* if $s_{F_1 \to F_2} \leqslant 2v_{\max}$. In order to test and correct for unrealistic connecting speed, we proceed as follows: We iterate on the collection of individual fragments by separating them into two subsets. First we consider the individual fragments whose last frame is less than the core of the first global fragment used for training (see section B.2.4), then the others. Let us consider a general individual fragment $F$ spanning frame numbers $[f_s, f_e]$. We check if there exist fragments $F_p$ and $F_n$ sharing the identity with $F$ and defined either in the past or in the future. If $F_p$ and $F_n$ do not exists, we proceed with the iteration. Otherwise, we evaluate the boundary speeds $s_{F_p \to F}$ and $s_{F \to F_n}$. We distinguish the following cases:

1. $s_{F_p \to F} > 2v_{\max}$ and $s_{F \to F_n} > 2v_{\max}$: If the identity of $F_p$ or $F_n$ is fixed or neither $F_p$'s consecutive previous fragment, nor $F_n$'s consecutive next fragments are unrealistic, we set $F$ to be reidentified.

2. $s_{F_p \to F} > 2v_{\max}$ and $s_{F \to F_n} \leqslant 2v_{\max}$: If the identity of $F_p$ is fixed $F$ is set to be reidentified. Otherwise $F_p$.

3. Symmetrically in the case $s_{F_p \to F} \leqslant 2v_{\max}$ and $s_{F \to F_n} > 2v_{\max}$.

**Reidentification of unrealistic consecutive fragments:** Let $F$ be an individual fragment to be reidentified. We compute the set $A$ of available identities by considering the identities of the individual fragments coexisting with $F$, and including the identity of $F$ itself. If $|A| = 1$, we assign the only available identity to $F$. Otherwise we proceed by calculating:

1. The subset $S \subseteq A$ of available identities that would not imply unrealistic boundary speeds given $F$.

2. $Q = \{\, i \in A \text{ s.t. } P_2\left(F, i\right) > \rho(F) \,\}$, where

$$\rho(F) = \begin{cases} \frac{1}{|F|} \text{ if } |F| > 1 \\ \frac{1}{n} \text{ otherwise} \end{cases},$$

   where $n$ is the number of tracked animals.

3. The set $C = Q \cap A$ of candidate identities, i. e. the set of identities that do not create duplications if assigned to $F$ and are at the same time acceptable with respect to both $P_2(F)$ and the speed model.

By considering the set $C$ just defined we have:

1. $|C| = 0$: No identities are available, thus $F$ is not identified.

2. $C = \{\, i \,\}$: We identify $F$ with $i$.

3. $|C| > 1$: $F$ is identified by considering the identity $i \in C$ realising the minimum boundary speed.

#### Crossing identification

Single individuals in a crossing are identified according to a python reimplementation of the algorithm described in [1. Supporting text, Section 2.12]. See idtracker.ai/postprocessing/ for the documentation and the source code of the algorithm.

### B.2.7 Output

In this section we discuss the final outputs of the algorithm: An estimation of the tracking accuracy will warn the user in case the algorithm could not proceed smoothly in the tracking process. The files containing the trajectories of each individual are saved and made available to the user.

#### Estimated accuracy

Let $\mathcal{I}$ be the set of all identified individual fragments, and $N = \sum_{F \in \mathcal{I}} |F|$ the total number of images in such fragments. We estimate the overall accuracy of the algorithm as

$$\mathcal{A} = \frac{\sum_{F \in \mathcal{I}} P_2(F, i) \cdot |F|}{N},$$

where $i$ is the identity assigned by the algorithm to the fragment $F$.

#### Individual trajectories

The algorithms outputs two individual trajectories files. One generated by considering the identification of individual images; the second by including the identification of the individual during crossings (see section B.2.6).

Both are organised as matrices with shape number of frames $\times$ number of individuals $\times$ 2, where the last two components are the position of the centroid of each individual in pixel coordinate, with respect to the entire frame.

## B.3 Human validation

After a video has been tracked, idtracker.ai provides an estimate of its own accuracy (see section B.2.7). Human validation is necessary to evaluate the goodness of the automatic accuracy assessment, notice recurrent inaccurate identifications, and evaluate the limit conditions in which the tracking system can work (e. g. fitness of the setup, suitability of the recording conditions and quality of the images).

We recall that the identity of an individual is maintained throughout individual fragments, thus a misidentification can happen only after a crossing or an occlusion. Notice that here a bad segmentation of the images (see section B.2.1) counts as an occlusion. Hence, the optimal validation would consist in evaluating that the identities of the animals before and after every crossing is conserved. Identities are assigned for the first time when labelling the individual fragments of the first global fragment used to train the idCNN. Hence, only by starting the validation from that global fragment, one can be sure that no switch of identities between two or more individuals occurred.

When dealing with large groups or particularly long videos, the validation of all the crossings is extremely costly. For this reason, we provide two procedures to facilitate this process. On the one hand, a global validation graphic interface allows to easily check the goodness of the identification of all the individuals in a segment of the video, correct their identities and compute the accuracy of the identification by considering the user-generated ground truth. On the other hand, an individual validation procedure allows to select a specific animal and follow it throughout the video. All the crossings, or occlusions that do not involve that individual are ignored, allowing a fast validation in long segments of the video.

### B.3.1 Global validation

Starting from the core of the first global fragment (see section B.2.4), we manually check that in every crossing all the identities of the animal involved in are maintained. Corrected identities are stored. After providing the segment $S = (\text{start - end})$ on which validation has been performed, we compute the following accuracy indices. Let $\mathcal{I}_S$ be the total number of individual images validated.

1. **Accuracy during protocol cascade**: Number of images correctly identified during the fingerpint protocol cascade, over the total number of individual images used to train the idCNN in $S$.
2. **Accuracy**: Number of images correctly identified over $\mathcal{I}_S$.
3. **Percentage of non-identified images**: Number of images non-identified, over $\mathcal{I}_S$.
4. **Percentage of misidentified images**: Number of images misidentified, over $\mathcal{I}_S$.

#### Individual validation

Individual validation is performed by considering a single individual at a time, and always proceeding from the core of the first global fragment used in the protocol cascade to the previous or the future frames. When validating the individual assigned to the identity $\iota$, we are interested only in crossings and occlusions in which it is involved. In this way, the validation is much faster and it is possible to control the quality of the identification in a wider timespan. After the correction of the misidentified images, we compute the accuracy in the assignment of $\iota$ by considering the number of correctly identified images, over the number of total images representing the individual.

# List of Symbols

| | |
|---|---|
| $A = \{ a_1, \ldots, a_n \}$ | Set collecting the elements $a_1, \ldots, a_n$. |
| $\lvert A \rvert$ | Number of elements in $A$. |
| $a \in A$ | The element $a$ belongs to the set $A$. |
| $A \cap B$ | Sets intersection: Set of the elements shared by $A$ and $B$. |
| $A \cup B$ | Sets union: Union of the elements of $A$ and $B$. |
| $A \setminus B$ | Difference between sets: Set of the elements of $A$ that are not elements of $B$. |
| s.t. | Such that. |