© 2018. Published by The Company of Biologists Ltd | Journal of Experimental Biology (2018) 221, jeb182469. doi:10.1242/jeb.182469

METHODS AND TECHNIQUES

UMATracker: an intuitive image-based tracking platform

Osamu Yamanaka[1,*,‡] and Rito Takeuchi[2,*]

ABSTRACT

Image-based tracking software are regarded as valuable tools in collective animal behaviour studies. For such operations, image preprocessing is a prerequisite, and the users are required to build an appropriate image-processing pipeline for extracting the shape of animals. Even if the users successfully design an image-processing pipeline, unexpected noise in the video frame may significantly reduce the tracking accuracy in the tracking step. To address these issues, we propose UMATracker (Useful Multiple Animal Tracker), which supports flexible image preprocessing by visual programming, multiple tracking algorithms and a manual tracking error-correction system. UMATracker employs a visual programming user interface, wherein the user can intuitively design an image-processing pipeline. Moreover, the software also enables the user to visualize the effect of image processing. We implement four different tracking algorithms to enable the users to choose the most suitable algorithm. In addition, UMATracker provides a manual correction tool for identifying and correcting tracking errors.

KEY WORDS: Animal behaviour, Computer vision, Object tracking, User interface, Visual programming

INTRODUCTION

In order to study animal behaviour, biologists have widely used image-based tracking software to extract animal trajectories from video data recorded by a fixed camera under laboratory conditions (Dell et al., 2014). Image-based tracking primarily involves two major steps: the preprocessing step and the tracking step.

At the preprocessing step, using image preprocessing (IP) techniques, animal shapes – termed blobs – are extracted from each frame in the video. Thus, the preprocessed video contains the animal shapes for each frame. This step comprises two sub-steps: first, each video frame is split into a foreground image (e.g. individual animals) and a background image (e.g. an experimental arena and inanimate objects); then, the noise due to unexpected incidents, such as floating faeces in a fish tank, dusty experimental conditions, etc., is removed from foreground images. These are accomplished by applying multiple IP algorithms, termed the image processing pipeline (IPP). In the tracking step, the tracking algorithm identifies animals in each video frame for a sequence of video frames using blobs. Then, a trajectory is constructed for each individual animal.

Designing an all-purpose IPP for use in a variety of experimental conditions is still a challenging task in the field of computer-vision applications. At present, users are required to develop appropriate IPPs for their experiments in order to extract blobs by using trial and error methods. Consequently, prerequisite understanding of the effect of each IP as well as the application of the software is required for users who are not familiar with IPs. Certain tracking software are preloaded with a user-interface (UI) for designing an IPP. For example, ImageJ, SwisTrack and Bonsai provide users with access to several IPs; however, the UI is rather complicated, preventing the user from intuitively using the appropriate IP. The inclusion of several IPs is advantageous as it enables the software to be used for a wide range of experimental conditions; however, identifying and applying the IPs in the complicated menu-driven UI is a major shortcoming of such software (Correll et al., 2006; Schneider et al., 2012; Lopes et al., 2015). In contrast, software such as Ctrax, ETHOWATCHER, idTracker, MouseMove and ToxTrac include only the IPs used by the respective developer. Moreover, users are allowed to adjust those parameters pertaining to each IP (Branson et al., 2009; Junior et al., 2012; Pérez-Escudero et al., 2014; Samson et al., 2015; Rodriquez et al., 2017). In other words, the software is easy to handle but place severe restrictions in terms of the experimental conditions. Thus, conventional software contain rich IPs with a complicated UI, or vice versa. Therefore, to address this shortcoming, the development of software with an intuitive UI with the added flexibility of selecting several IPs is required.

It should be noted that even if the users succeed in designing an appropriate IPP and run a tracking algorithm, the obtained trajectory data may have tracking errors. The tracking errors occur under experimental conditions as a result of animals frequently overlapping each other, reflection of light from a water surface or experimental equipment which hides the animal blob. In general, the accuracy of the trajectory is reduced because of the occurrence of two tracking errors: ‘identity-losing’ errors and ‘identity-swapping’ errors. ‘Identity-losing’ errors occur when an inanimate object in a video frame is recognized as an animal; if an inanimate object is identified in the previous frame and exists in the current frame, then an incorrect position is recorded as the animal position. ‘Identity-swapping’ errors occur when several animals are present in the same frame, and the identity of one animal is misallocated to another animal. Such tracking errors tend to propagate to the subsequent frames. Moreover, each tracking algorithm has its associated advantages and disadvantages (Smeulders et al., 2014); thus, the users have to select an appropriate image-based tracking algorithm based on the abovementioned trade-offs.

Tracking accuracy can be improved by using additional IPs and adjusting the parameters of each IP through trial and error. However, it is necessary to repeatedly run a tracking algorithm to confirm the trajectory of the animal in the video frames. Thus, the task of automatically tracking animals by taking into account the adjustments made to an IPP and running a tracking algorithm multiple times is a tedious one. Therefore, to save time, users are required to manually correct the tracking errors during and after the tracking step.

[1]Department of Mathematical and Life Sciences, Graduate School of Science, Hiroshima University, Higashi-Hiroshima, Hiroshima 739-8526, Japan. [2]Yahoo Japan Corporation, Tokyo 102-8282, Japan.
*These authors contributed equally to this work

‡Author for correspondence (oyamanaka@mitou.org)

O.Y., 0000-0002-3733-1693; R.T., 0000-0002-2672-845X

Received 11 April 2018; Accepted 21 June 2018

Hence, the requirements of the image-based tracking software can be summarized as follows. (1) A user without expertise should be able to construct an appropriate IPP while understanding the effect of each IP. (2) A user should be able to choose a tracking algorithm from multiple tracking algorithms. (3) A user should be able to rectify tracking errors manually and directly when errors occur.

MATERIALS AND METHODS

The UMATracker comprises three units: FilterGenerator, Tracking and TrackingCorrector. Each unit corresponds to each process of the tracking analysis: preprocessing of video images, object tracking and manual correction, respectively. The reader is advised to see Movies 1, 2 and 3 for additional details.

Preprocessing step (FilterGenerator)

At the preprocessing step, the user designs an IPP on the editor panel using the visual programming environment Blockly (Google Inc., CA, USA), wherein each IP is represented as a block (Fig. 1A). FilterGenerator includes necessary and sufficient IP blocks (e.g. morphology algorithms for noise reduction, thresholding image binarization algorithms and a region selector to exclude obstacles).

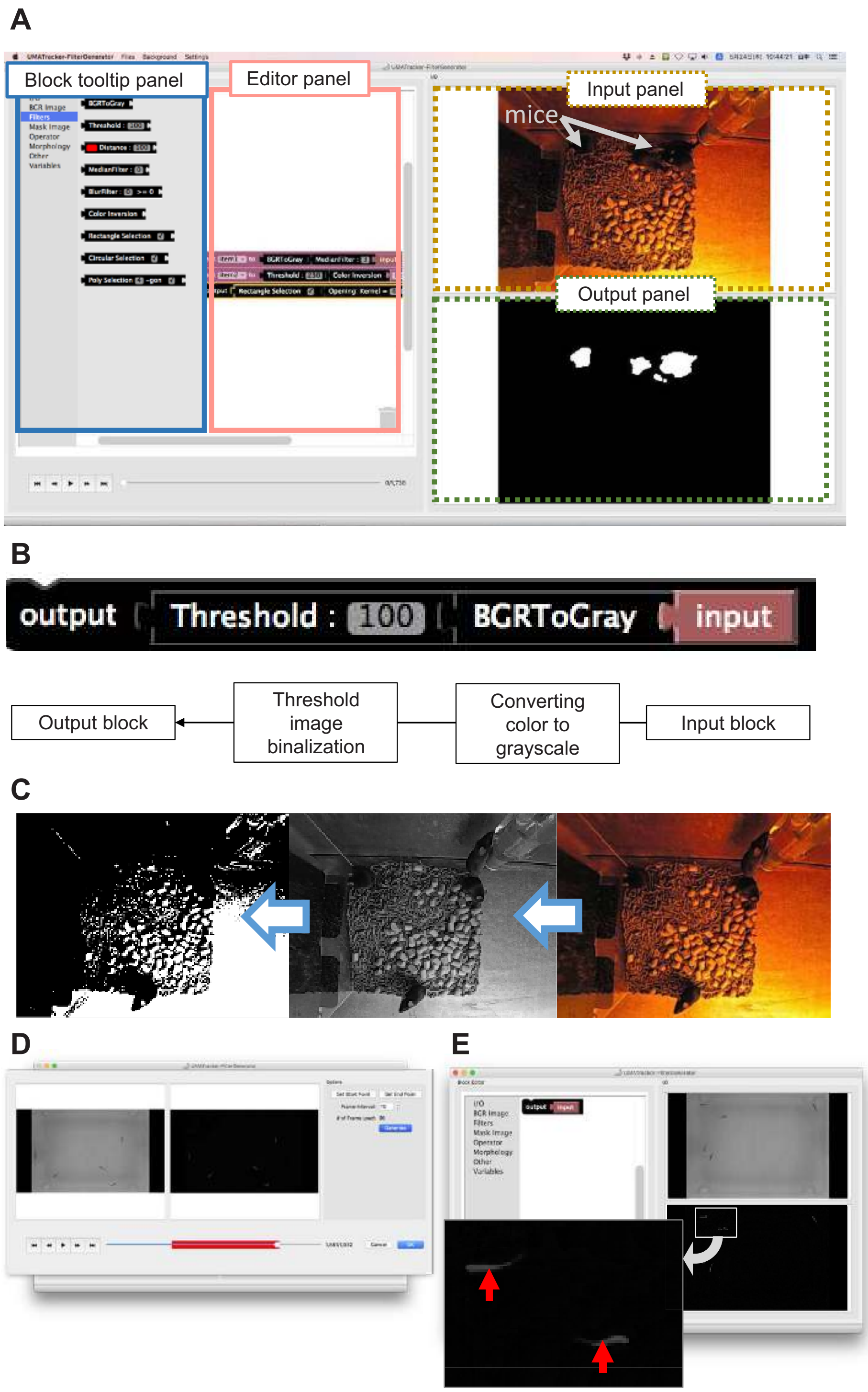

Fig. 1. Features of the FilterGenerator system. (A) The FilterGenerator user interface. The left viewport is the image processing pipeline (IPP) editor. The assembled block represents an IPP. The right viewport shows the original input video (input panel) and the output video obtained by applying the IPP (output panel). (B) An IPP showing conversion of colour to gray block and a threshold binarization block. (C) Results of image processing by the IPP in B. (D,E) Example of statistical background subtraction. The original video is from Pérez-Escudero et al. (2014). (D) User interface for statistical background subtraction. (E) The result of background subtraction. Each red arrow indicates an extracted animal shape.

These enable the software to be used for a wide range of experimental conditions.

By connecting the ‘input’ block, which corresponds to each input video frame, to a series of IP blocks which describes a specific IPP, each IP algorithm sequentially applies to the input video frame (Fig. 1B,C). Then, if an assembled block is connected to the ‘output’ block, the video frame converted by the IPP blocks is displayed on ‘output panel’ (Fig. 1A). In order to enable visualization of the effect of an IP, when the user moves the mouse cursor to the IP block of the block tool tip panel, the video frame converted by its IP is displayed on the output panel. Thus, the user can interactively design an IPP while ensuring that the desired effect of an IP is obtained.

As shown in Fig. 1B, to use the threshold image binarization, the user can design an IP that combines a converting colour to gray block and a threshold binarization block. Subsequently, the user adjusts the threshold value on the threshold binarization block to confirm the converted video frame on the output panel.

In cases where the animals cannot be distinguished from the experimental background, a background subtraction algorithm can be applied (Stauffer and Grimson, 1999), which detects the background image using the video frame at the frame interval between the start point time and the end point time input on a background subtraction dialogue (Fig. 1D,E). Then, the user can design an IPP similar to the threshold binarization (as mentioned previously).

Tracking step (Tracking)

At the tracking step, the user initially sets the number of animals in a video frame. Then, the user clicks the ‘play’ button to run a tracking algorithm. The user can monitor the detected position and the

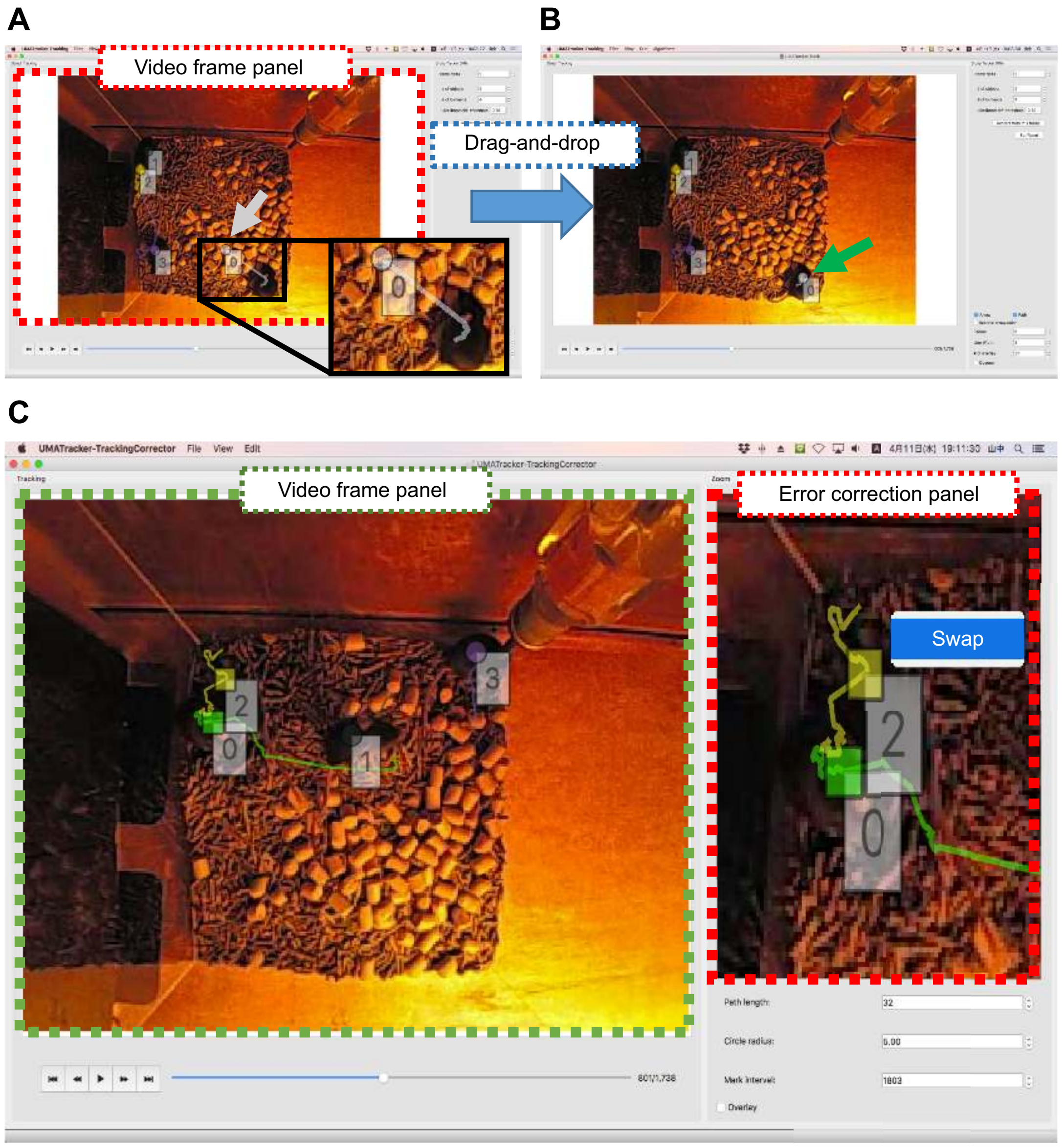

Fig. 2. Error correction in Tracking and TrackingCorrector. (A) Tracking user interface. The video frame panel shows a video and the obtained trajectory data. The white arrow indicates where the identity-losing error occurred. (B) Scene after manual identity-losing error correction by dragging the position indicator (white circle) and dropping it onto the correct position indicated by the green arrow. (C) TrackingCorrector user interface. When a user selects an area on the video frame panel, the expanded region is displayed on the error correction panel. The user can confirm the validity of trajectories and fix identity-swapping errors by selecting the two relevant trajectories from the error correction panel.

trajectory of the animals sequentially as displayed in the video frame.

When the user detects an identity-losing error, the user may choose to pause the tracking algorithm by clicking the ‘stop’ button. To rectify the error, the user drags the position indicator and drops it on the correct position (Fig. 2A,B).

The proposed software supports the implementation of four tracking algorithms: RMOT (Yoon et al., 2015), Optical Flow DualTVL1 (Zach et al., 2007; Sánchez Pérez et al., 2013), K-means (Lloyd, 1982; Arthur and Vassilvitskii, 2007) and GroupTracker (Fukunaga et al., 2015).

Rectifying identity-swapping errors (TrackingCorrector)

At this step, if the user detects an identity-swapping error while monitoring the trajectories of the animals detected in the tracking step, the software allows the user to select the region where the tracking error occurred (from the left viewport of the application window in Fig. 1A), and the expanded region is displayed in the right viewport (Fig. 2C). Then, the user selects two points that represent individual animals, and clicks on the ‘swap’ button. In this way, the identity-swapping error is corrected.

RESULTS AND DISCUSSION

We have developed UMATracker, user-friendly software that offers a platform to run image-based tracking algorithms. The stable version of the software and source code can be accessed from the GitHub repository at http://ymnk13.github.io/UMATracker/. In addition, an instruction manual is provided at https://umatracker.github.io/UMATracker-manual-en/. There are two conditions for using UMATracker. First, the user must cite this work if the user publishes a scientific achievement using trajectory data obtained using the software. Second, the user must cite corresponding papers on related tracking algorithms because the tracking algorithms have been developed by scientists in the field of computer vision.

UMATracker has been applied in biology research as well as for educational purposes (Yokoi et al., 2016; Shimoji et al., 2017preprint; Mizumoto and Dobata, 2018preprint). Users have applied UMATracker to various animal species, such as mice and laboratory rats, pigeons, goldfish, sea bream, squid, termites and ants (Fig. 3).

In comparison with conventional tracking software, UMATracker has three main features: a flexible IPP through visual programming (FilterGenerator), multiple tracking algorithms (Tracking) and an intuitive manual error-correction system (TrackingCorrector).

The visual programming environment for designing an IPP is more intuitive for biologists than a traditional menu-driven UI (Koelma and Smeulders, 1994). However, it is disadvantageous to manually design an IPP because the users have to design an appropriate IPP. To enable novice users to easily use UMATracker, the Slack forum has been created in order to allow users to seek guidance from experts.

In the proposed software, four tracking algorithms were implemented, thereby enabling the user to select the appropriate algorithm corresponding to the experimental situation and the animal species. As experimental features, Tracking has algorithms which extract detailed information for each animal shape, e.g. the head direction, the body shape or the skeleton. Moreover, to improve the versatility of the software, developers can add tracking algorithms as a plug-in.

The proposed software enables the user to manually correct an animal’s position at the instance where the tracking errors occur.

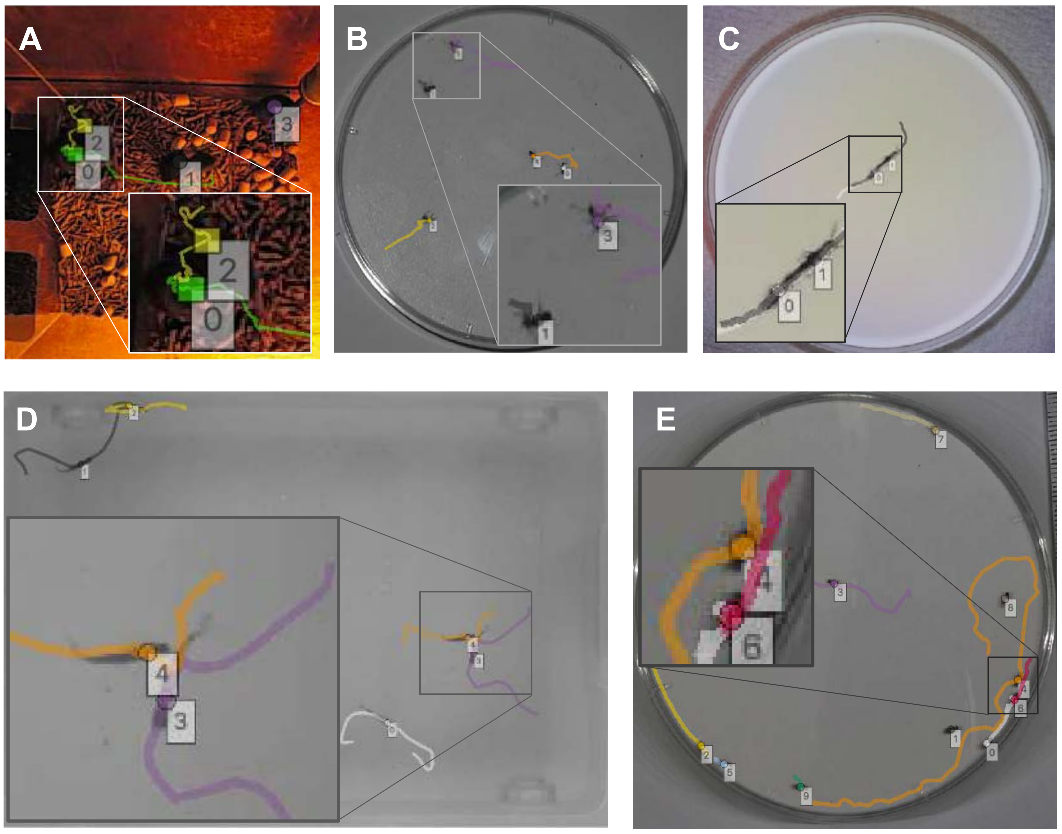

Fig. 3. Examples of trajectory data obtained using UMATracker. (A) Four mice in a non-uniformly illuminated cage. (B) Five ants on a plate. (C) Two termites on a plate. (D) Five zebrafish (video is from Pérez-Escudero et al., 2014). (E) Fifteen fruit flies on a plate.

This reduces the propagation of tracking errors across video frames, thereby avoiding laborious and time-consuming operations.

In this study, we focused on cases where the animals move only in the experimental arena. Therefore, the software cannot accurately track multiple animals when an animal goes out of/disappears from a video frame. Therefore, to process videos wherein the animals move in and out of the video frame, the user is required to divide the video sequence into videos wherein the number of animals in each video is constant.

Conventional software allows the implementation of batch processing, wherein a large number of video files are processed simultaneously. However, UMATracker does not support batch processing because tracking errors occur frequently under certain environmental conditions, and the user has to confirm and fix such errors immediately.

Acknowledgements
We express our sincere thanks to Dr Yuzuru Ikeda and Dr Chikatoshi Sugimoto (Department of Marine and Environmental Sciences, Graduate School of Engineering and Science, University of the Ryukyus) for granting us the opportunity to undertake this work. We thank Dr Masataka Goto (National Institute of Advanced Industrial Science and Technology), Dr Masashi Shiraishi, Dr Hiraku Nishimori (Graduate School of Science, Hiroshima University) and Dr Kentaro Ito (Department of Frontier Bioscience, Hosei University) for their useful and insightful discussions. We would also like to thank Mr Teppei Kuraya, Dr Kosuke Sawa (Department of Psychology, School of Human Sciences, Senshu University), Dr Nobuaki Mizumoto (School of Life Science, Arizona State University) and Dr Takahiro Chihara (Department of Biological Science, Graduate School of Science, Hiroshima University) for providing sample videos.

Competing interests
The authors declare no competing or financial interests.

Author contributions
Conceptualization: O.Y., R.T.; Methodology: O.Y., R.T.; Software: O.Y., R.T.; Writing - original draft: O.Y., R.T.; Writing - review & editing: O.Y., R.T.; Visualization: O.Y., R.T.; Supervision: O.Y., R.T.; Project administration: O.Y., R.T.

Funding
This software development was supported by the Exploratory IT Human Resources Project (The MITOU Program) of Information-Technology Promotion Agency (IPA), Japan in the fiscal year 2015.

Data availability
The software and source code developed here, and the associated instruction manual can be accessed from the GitHub repository: http://ymnk13.github.io/UMATracker/; https://umatracker.github.io/UMATracker-manual-en/

Supplementary information
Supplementary information available online at
http://jeb.biologists.org/lookup/doi/10.1242/jeb.182469.supplemental

References

Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. Proc. ACM-SIAM SODA 2007, 1027-1035.
Branson, K., Robie, A. A., Bender, J., Perona, P. and Dickinson, M. H. (2009). High-throughput ethomics in large groups of Drosophila. *Nat. Methods* **6**, 451-457.
Correll, N., Sempo, G., De Meneses, Y. L., Halloy, J., Deneubourg, J. L. and Martinoli, A. (2006). SwisTrack: a tracking tool for multi-unit robotic and biological systems. Proc. IEEE/RSJ IROS 2006, 2185-2191.
Dell, A. I., Bender, J. A., Branson, K., Couzin, I. D., de Polavieja, G. G., Noldus, L. P. J. J., Pérez-Escudero, A., Perona, P., Straw, A. D., Wikelski, M. et al. (2014). Automated image-based tracking and its application in ecology. *Trends Ecol. Evol.* **29**, 417-428.
Fukunaga, T., Kubota, S., Oda, S. and Iwasaki, W. (2015). GroupTracker: video tracking system for multiple animals under severe occlusion. *Comput. Biol. Chem.* **57**, 39-45.
Junior, C. F. C., Pederiva, C. N., Bose, R. C., Garcia, V. A., Lino-de-Oliveira, C. and Marino-Neto, J. (2012). ETHOWATCHER: validation of a tool for behavioral and video-tracking analysis in laboratory animals. *Comput. Biol. Med.* **42**, 257-264.
Koelma, D. and Smeulders, A. (1994). A visual programming interface for an image processing environment. *Pattern Recogn. Lett.* **15**, 1099-1109.
Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Trans. Inform. Theory* **28**, 129-137.
Lopes, G., Bonacchi, N., Frazão, J., Neto, J. P., Atallah, B. V., Soares, S., Moreira, L., Matias, S., Itskov, P. M., Correia, P. A. et al. (2015). Bonsai: an event-based framework for processing and controlling data streams. *Front. Neuroinform.* **9**.
Mizumoto, N. and Dobata, S. (2018). Adaptive switch to sexually dimorphic movements by partner-seeking termites. *bioRxiv*
Pérez-Escudero, A., Vicente-Page, J., Hinz, R. C., Arganda, S. and de Polavieja, G. G. (2014). idTracker: tracking individuals in a group by automatic identification of unmarked animals. *Nat. Methods* **11**, 743-748.
Rodriquez, A., Zhang, H., Klaminder, J., Brodin, T., Andersson, P. L. and Andersson, M. (2017). ToxTrac: a fast and robust software for tracking organisms. *Methods Ecol. Evol.* **9**, 1-5.
Samson, A. L., Ju, L., Kim, H. A., Zhang, S. R., Lee, J. A. A., Sturgeon, S. A., Sobey, C. G., Jackson, S. P. and Schoenwaelder, S. M. (2015). MouseMove: an open source program for semi-automated analysis of movement and cognitive testing in rodents. *Sci. Rep.* **5**.
Sánchez Pérez, J. S., Meinhardt-Llopis, E. and Facciolo, G. (2013). TV-L1 optical flow estimation. *IPOL* **3**, 137-150.
Schneider, C. A., Rasband, W. S. and Eliceiri, K. W. (2012). NIH Image to ImageJ: 25 years of image analysis. *Nat. Methods* **9**, 671-675.
Shimoji, H., Mizumoto, N., Oguchi, K. and Dobata, S. (2017). Caste-biased movements by termites in isolation. *bioRxiv.*
Smeulders, A. W., Chu, D. M., Cucchiara, R., Calderara, S., Dehghan, A. and Shah, M. (2014). Visual tracking: an experimental survey. *IEEE Trans. Pattern Anal.* **36**, 1442-1468.
Stauffer, C. and Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. *Proc. 1999 IEEE CVPR.* **2**, 246-252.
Yokoi, S., Ansai, S., Kinoshita, M., Naruse, K., Kamei, Y., Young, L. J., Okuyama, T. and Takeuchi, H. (2016). Mate-guarding behavior enhances male reproductive success via familiarization with mating partners in medaka fish. *Front. Zool.* **13**.
Yoon, J. H., Yang, M. H., Lim, J. and Yoon, K. J. (2015). Bayesian multi-object tracking using motion context from multiple objects. Proc. 2015 IEEE WACV. 33-40.
Zach, C., Pock, T. and Bischof, H. (2007). A duality based approach for realtime TV-L 1 optical flow. *Pattern Recogn.* **4713**, 214-223.