Methods in Ecology and Evolution

BRITISH ECOLOGICAL SOCIETY

Methods in Ecology and Evolution 2016 doi: 10.1111/2041-210X.12702

APPLICATION

ClickPoints: an expandable toolbox for scientific image annotation and analysis

Richard C. Gerum*,†,[1], Sebastian Richter†,[1], Ben Fabry[1] and Daniel P. Zitterbart[1,2,3]

[1]*BioPhysics Group, University of Erlangen-Nürnberg, Erlangen, Germany;* [2]*Applied Ocean Physics and Engineering, Woods Hole Oceanographic Institution, Woods Hole, MA, USA; and* [3]*Alfred-Wegener-Institute, Helmholtz-Center for Polar and Marine Research, Bremerhaven, Germany*

Summary

1. Optical recordings are ubiquitous in current scientific research. As recording and storing images and videos have become simpler and cheaper over the past years, new and more efficient tools are needed to analyse the growing amount of data. While current scientific image analysis tools focus on medical 3D data sets, a tool for efficiently handling large-scale 2D time-series recordings is still missing.

2. We developed *ClickPoints*, an expandable, open-source, Python-based software to fill this gap. *ClickPoints* combines and streamlines the three main steps of image analysis – visualization, annotation and evaluation – in one program.

3. *ClickPoints* enables the user to (i) efficiently review large data sets of images and videos, (ii) annotate interesting findings and (iii) facilitate manual and automatic evaluations. *ClickPoints* is highly versatile, ranging from clicking and selecting objects with simple markers, drawing masks and computing trajectories, up to (iv) custom-written Python add-ons for adapting or extending the available features. These add-ons (v) reduce development time by utilizing *ClickPoints*' display, interface and storage solutions. In addition, *ClickPoints* provides an (vi) extensive data base application programming interface to facilitate efficient storage and retrieval of results.

4. *ClickPoints* is designed to simplify repetitive and labour-intensive scientific tasks, especially when dealing with large image and video time-series recordings. *ClickPoints* offers cross-platform support for Windows and Linux and is released under the GPLv3 license. Download, documentation and numerous examples are available at http://clickpoints.readthedocs.io.

Key-words: ground truth, image annotation, image evaluation, labelling, open source, Python, time-series data, visual data processing

Introduction

Since the invention of CCD chips, optical data recordings have gained increasing importance in almost every area of scientific research (Janesick 2001) ranging from microscopy to astronomy. The time span of recordings ranges over many orders of magnitude, ranging from high speed cameras (Ballarotti, Saba & Pinto 2005; Dhillon *et al*. 2007) for detecting ultrafast processes at the scale of microseconds to time-lapse recordings (Reif & Tornberg 2006; Egea *et al*. 2011) for detecting slow processes at the scale of days, months or years. With the growing availability of digital cameras (Fossum & Hondongwa 2014) and inexpensive hard drives (Kryder & Kim 2009), our ability to record large amounts of images and videos increases steadily, but our ability to handle and process them severely lags behind.

Image data have to be visualized for the user to select and annotate interesting or useful segments for further evaluations. But up to now, no comprehensive software tool exists that can handle these three fundamental steps to evaluate large time-series data sets: visualization, annotation, evaluation. Therefore, we developed *ClickPoints*, an expandable program to combine and streamline the tasks of viewing, annotating and evaluating visual recordings in one single program, aiming to provide a time-efficient alternative to existing solutions.

Common display solutions for videos (e.g. VLC Player, VideoLan Organization 2016) and images (e.g. IrfanView, Skiljan 2016) do not focus on scientific needs. By contrast, *ClickPoints* extends basic display functionality for scientific applications. For example, *Clickpoints* can display an arbitrary number of videos and images as a seamless time series and provides enhanced features for playback, navigation and annotation.

Well known scientific image analysis tools, such as ImageJ (Schneider, Rasband & Eliceiri 2012), Fiji (Schindelin *et al*. 2012), BioImageXD (Kankaanpää *et al*. 2012) or Icy (de Chaumont *et al*. 2012) focus on microscope images and 3D reconstruction. *ClickPoints* specializes on the handling of 2D time-series data sets and aims to provide an intuitive graphical

*Correspondence author. E-mail: richard.gerum@fau.de
†These authors contributed equally to this work.

display, user interface, communication Application Programming Interface (API) and storage API to encourage the users to develop new add-ons from scratch or to adapt existing add-ons to their application.

ClickPoints is designed to efficiently view, annotate and evaluate time-series data sets, ranging from simple tasks to the development of complex evaluation tool chains.

Design and implementation

ClickPoints is written in Python (van Rossum 1995), a widely-used free and open-source high-level interpreted programming language. The user interface is implemented using the Qt framework (The Qt Company 2016) to ensure cross platform compatibility. The backend is based on ImageIO (Klein 2014), supporting a large range of video or image formats, including user defined custom formats.

VIEWING DATA

An important feature of *ClickPoints* is its advanced image and video display capabilities, designed for screening data ranging from high-resolution giga-pixels stitched images to sequences containing millions of images. *ClickPoints* is capable of handling more than 100 different image and video formats, supports recursive loads and seamless concatenation of multiple video and image files. All opened files are treated as one continuous stream of frames, which can be displayed at varying speed or with skipped frames, to get a quick overview of a data set or to visually investigate dynamic processes at different timescales. The user can seamlessly zoom in and out, rotate and pan the image during playback and thus focus on different details of the scene. In addition, frames can be contrast-enhanced or gamma-adjusted to reveal faint details (Fig. 1B).

ClickPoints provides various tools for navigating through a data set. Frames can be skipped in defined intervals using key shortcuts, and can be accessed by a frame-based or timestamp-based slider (timeline) (Fig. 1A) that uses time information extracted from the data set. The timeline can be panned or zoomed in and out from seconds to years to visualize the distribution of available frames. Start and end of playback can be specified to loop over frames of interest.

ANNOTATING FRAMES

ClickPoints can add text annotations to the currently displayed frame (Fig. 1F) to mark interesting events. Annotations can contain free text, multiple custom-defined tags, or a numerical quality rating. Frames with an annotation are marked in the frame slider for easy retrieval. The annotation browser (Fig. 1G) gives an overview over all annotations and allows for quick access to the annotated events. Annotations are especially useful for large data sets; they enable fast navigation within the data set, and they mark and tag events of interest, which can then be used as starting point for further evaluations.

SELECTING POINTS AND REGIONS OF INTEREST

ClickPoints allows the user to mark points and region of interest. The user can specify different types, categories and colours to mark single points, lines, rectangles or tracks over multiple

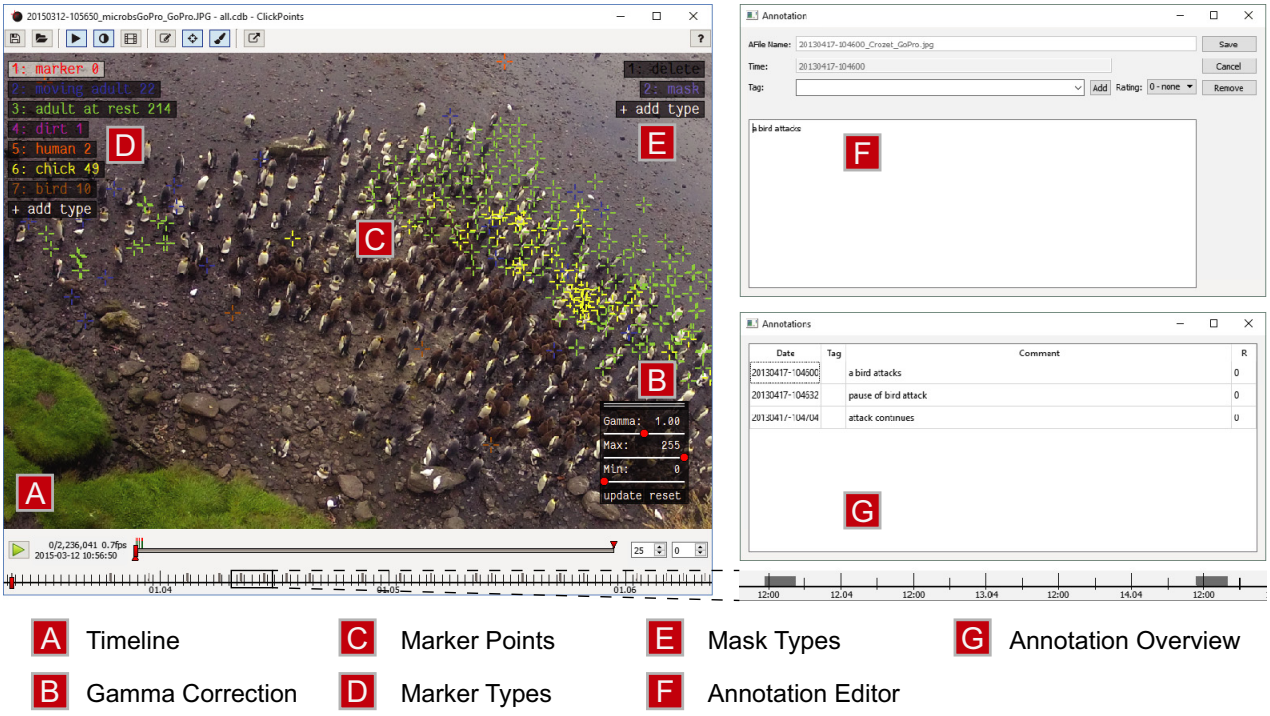


Fig. 1. The *ClickPoints* interface. *ClickPoints* displays the images in a frame-based and (if applicable) a timestamp-based timeline (A), allows for gamma corrections (B), and the addition of markers (C, D), masks (E) and annotations (F, G).

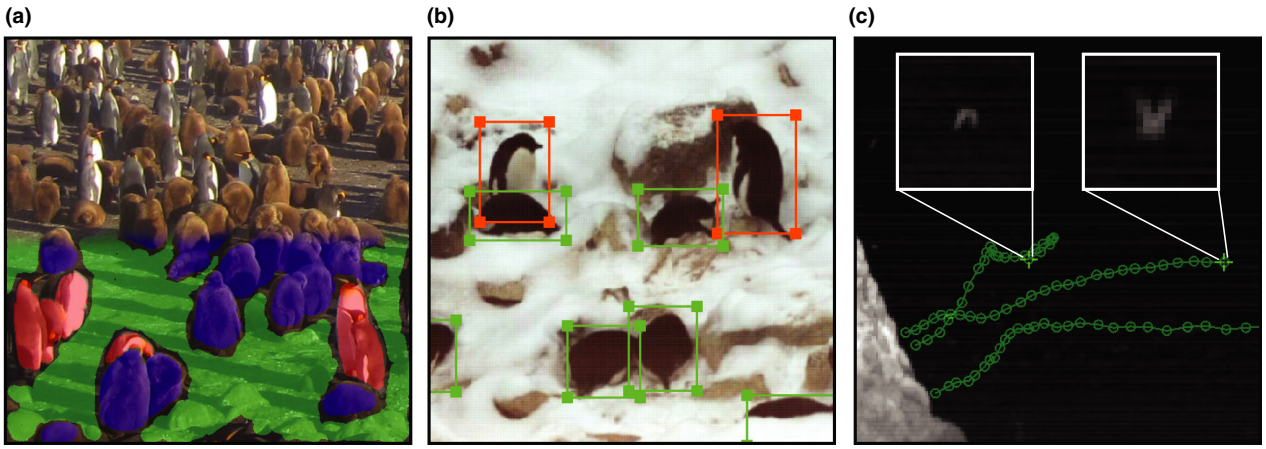


Fig. 2. Annotations for Machine Learning applications. (a) Manually drawn masks for training a neural-network to segment an image (upper part) into: background (green), juvenile King Penguin (blue), and adult King Penguin (red) classes (lower part of the image). (b) Adélie Penguins selected with *ClickPoints* to train an object detection algorithm (orange rectangles: standing penguins, green rectangles: lying penguins). (c) Ground truth tracks to evaluate the performance of a thermal imaging-based bird tracking algorithm.

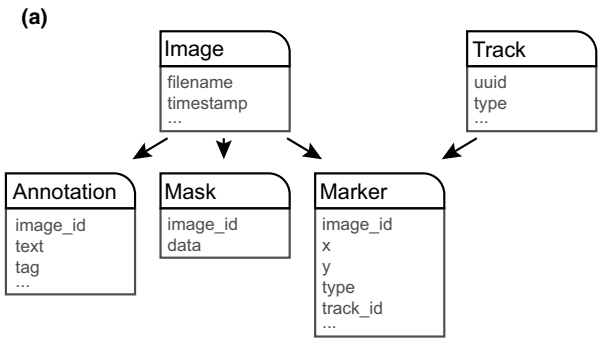
frames (Fig. 1C–D). For selecting arbitrary regions, the user can draw masks with different colours with adjustable display transparency (Fig. 1E). Markers and masks provide the basis for subsequent manual image evaluations, e.g. point counts, positions, line lengths, rectangle sizes etc., or can be used as input for further software components in the evaluation tool chain, e.g. for tracking or machine learning algorithms (Fig. 2).

DATA STORAGE AND ACCESS

The data base architecture and its ease of access through the API and the Structured Query Language (SQL) are a core feature of *ClickPoints*.

An SQLite data base is used as the application file format to store all data. While this may seem more complicated than a simple text or spreadsheet file, it offers numerous advantages. Everything is stored in a single file, which can be accessed through a high level interface provided by SQL. The file structure, consisting of multiple SQL tables (Fig. 3a), provides all necessary information to access the data and understand its structure. An SQL query can, for example, retrieve a list of frames, all marked points for one frame, or all points of one track spanning multiple frames (Fig. 3b). This is significantly faster than reading multiple text files and searching for the required data. SQLite is available for all major programming languages, e.g. Python, C++, Java, Matlab, R etc., enabling the user to choose a favourite language to access and further evaluate the data. In addition, the user may extend the data base to store custom information alongside the *ClickPoints* data. As multiple processes can access the data base at the same time, evaluation scripts can run concurrently to the *ClickPoints* instance and therefore directly display results.

ClickPoints comes with a Python API to comfortably access the data base even without any knowledge of SQL. Detailed examples on how to use *ClickPoints* and the API are available for download at http://clickpoints.readthedocs.io.



(b)

Task	SQL	CP AP
Retrieve all images	SELECT * FROM image	db.getImages()
Retrieve every marker of image number 5	SELECT * FROM marker WHERE image_id == 5	db.getMarkers(image = 5)
Retrieve marker coordinates of a track	SELECT x,y FROM marker WHERE track_id == 2	db.getTrack(id = 2).points

Fig. 3. Data base structure and access. (a) Simplified *ClickPoints* structured query language table schema. (b) Example tasks and corresponding structured query language or application programming interface commands.

SUPERVISED AUTOMATED EVALUATION

ClickPoints provides an interface to run add-on scripts and offers bidirectional communication between the script and the *ClickPoints* instance. The script can access information from the data base, write results to the data base, and force *ClickPoints* to update its display or change its current frame. *ClickPoints* can launch and stop these scripts and share its data through the data base.

This add-on interface was developed for the implementation of semi-automated evaluations, as most algorithms eventually need some user interaction. User input to start an algorithm, e.g. a starting point for subsequent tracking, can be provided via the *ClickPoints* interface. The evaluation

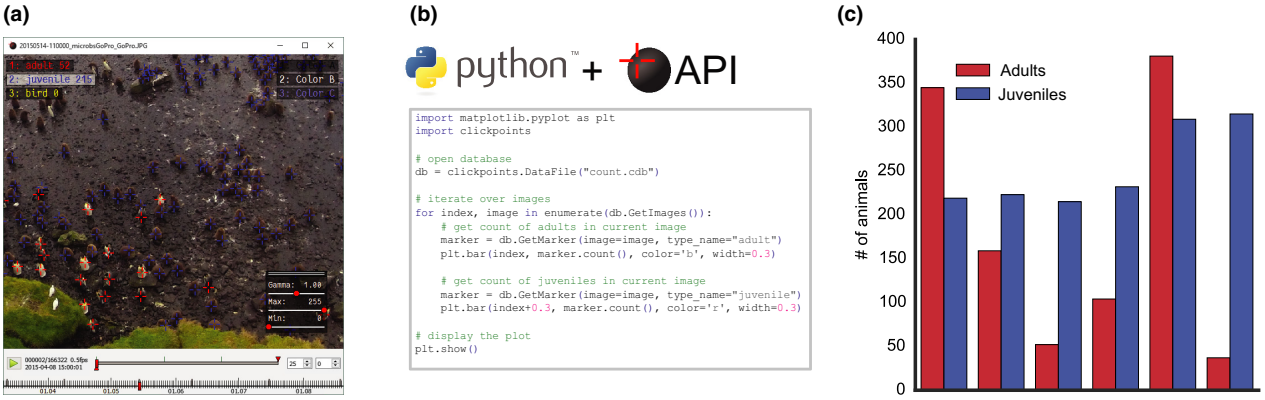


Fig. 4. Counting of different types of animals. (a) Marked adult and juvenile penguins. (b) Python code to access and display the data, using the application programming interface. (c) Counts of adults and juveniles in the analysed images.

results are directly displayed in *ClickPoints* to facilitate immediate review and potential correction by the user. This saves time and efforts to redevelop dedicated input and display routines for each algorithm.

This approach extends *ClickPoints* far beyond its basic feature set and allows for a broad range of advanced and specialized applications.

Example applications

In the following section, we demonstrate the use of *ClickPoints* for a variety of scientific image evaluation tasks. The first two examples demonstrate basic functionality of labeling data and extracting the results with the help of the API. Further examples show the extension of the basic *ClickPoints* functionality by add-ons and its interaction with external programs. All examples are taken from ongoing scientific studies, however, the relevance of the provided results are not within the scope of this manuscript and therefore are not discussed in detail.

COUNTING ANIMALS IN DIFFERENT CATEGORIES

In this example, we demonstrate how *ClickPoints* can be used to count objects of different types in an image.

Image recordings of King Penguins (*Aptenodytes patagonicus*) are used to evaluate the abundance of adult and juvenile animals over time. Images were obtained by a GoPro Hero 2 camera, located at the Baie du Marin King penguin colony on Possession Island of the Crozet Archipelago (French Polar Institute Paul-Emile Victor (IPEV) - Program 137 (CNRS/CSM) Le Bohec 2013).

Images are loaded into *ClickPoints*, and marker types are defined for the two categories (juvenile vs. adult). Penguins are marked by clicking on the animal while the appropriate category is selected. Categories can be selected or switched quickly by clicking on the corresponding button or by pressing a number key. Markers can be moved or deleted if necessary. For better visualization and accuracy, the images can be zoomed and panned using the mouse. In our example, all penguins in one area of the colony were labelled in six consecutive time-lapse images (Fig. 4a). Data access and a simple display of the results (Fig. 4c) can be achieved with nine lines of Python code, using the API (Fig. 4b). The resulting marker counts of the different categories, displayed with matplotlib (Hunter 2007), show how many adults and juveniles were present in each of the six images (Fig. 4c).

The example data set and evaluation code can be found as example "Count Penguins" at: http://clickpoints.readthedocs.io/en/latest/example_countpenguins.html.

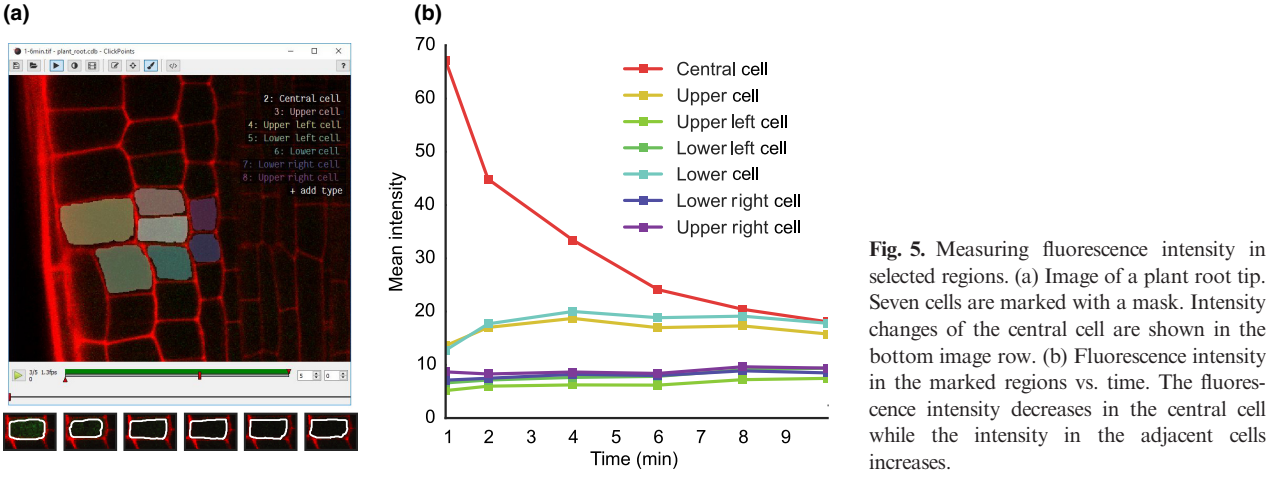


Fig. 5. Measuring fluorescence intensity in selected regions. (a) Image of a plant root tip. Seven cells are marked with a mask. Intensity changes of the central cell are shown in the bottom image row. (b) Fluorescence intensity in the marked regions vs. time. The fluorescence intensity decreases in the central cell while the intensity in the adjacent cells increases.

AREA-BASED PROTEIN FLUORESCENCE ANALYSIS IN PLANT ROOTS

This example serves to demonstrate how the mask drawing feature of *ClickPoints* can be used to analyse fluorescence intensity profiles and dynamics in microscope images.

Images of an *Arabidopsis thaliana* root tip, obtained using a two-photon confocal microscope, were recorded at 1 min time intervals (Gerlitz 2016). The plant roots expressed a photoactivatable green fluorescent protein, which after activation with a UV pulse diffuses from the activated central cell to the neighbouring cells. A mask type is defined for each cell of interest, and an arbitrary shaped mask is drawn over the area of each cell in every time step (Fig. 5a). The temporal evolution of the average fluorescence pixel intensities of each cell indicates how the green fluorescent protein diffuses from one cell to the adjacent cells (Fig. 5b). The drawn masks are stored in the data base for later review, modification, and extraction of additional information, e.g. the cell area or the standard deviation of fluorescence intensities.

The example data set and evaluation code can be found as example "PlantRoot" at: http://clickpoints.readthedocs.io/en/latest/example_plantroot.html.

IMAGE STABILIZATION OF EMPEROR PENGUIN RECORDINGS

This example shows how *ClickPoints* can be used to remove drift and vibration artefacts from image sequences and to manually track animals in the corrected images.

An Emperor Penguin (*Aptenodytes forsteri*) colony was recorded with an Allied Vision GE4000C (11 Mpx) camera and a 400 mm lens (Richter 2011). The average wind speed during the recordings was 22 km h^{-1}, which causes significant camera vibrations. To remove the vibrations, the add-on "DriftCorrection" was used to track the movements of immobile features in the images, which are marked with a rectangular region of interest (ROI). The ROI is detected in the subsequent images, using a cross-correlation template matching algorithm (Bradski 2000) with subpixel accuracy.

The wind-induced frame-to-frame displacements of the ROI coordinates with respect to its coordinates in the first frame are stored as offset (Fig. 6c) in the data base. *ClickPoints* can use these offsets for a stabilized display of images and markers, without any additional computational cost (Fig. 6a). Therefore, it is not necessary to save shifted copies of the images. Image stabilization is crucial for the extraction of correct trajectories (Fig. 6a and b) in the presence of vibrations or drift.

SUPERVISED TRACKING OF FIDUCIAL MARKERS IN MAGNETIC TWEEZER MEASUREMENTS

In this example, we demonstrate how *ClickPoints* can be extended with an add-on for supervised tracking of fiducial marker beads attached to a cell.

We analyse the cell deformations in response to a force applied with a magnetic tweezer (Kollmannsberger & Fabry 2011). A 5 μm paramagnetic bead coated with extracellular matrix proteins is bound to the surface of a cell. A needle-shaped tip with an electromagnet is then used to apply defined forces to this bead. In addition, smaller (1 μm) non-magnetic beads were also attached to the cell. These smaller beads serve as fiducial markers of intracellular movements (Bonakdar *et al.* 2016). To extract the displacement of the fiducial marker beads, the images were loaded in *ClickPoints,* and the starting positions of the beads were marked.

The add-on takes these user-defined points as starting points and tracks the movement of the marked objects over the subsequent images, using a sparse Lucas-Kanade optical flow algorithm (Bradski 2000; Bouguet 2001). *ClickPoints* then displays the tracking results for each frame. The user can interrupt and resume the algorithm at any point, e.g. if a tracking error occurs. Marker positions can be manually corrected to serve as new starting points when tracking is resumed. The power of this approach lies in the easy visualization and correction of tracking results, as any computer vision algorithm will eventually yield erroneous results that require user intervention.

The example tracking algorithm is implemented in <50 lines of code and has been adapted to track penguins, pollen tube growth and natural killer cells. This add-on takes full advantage of *ClickPoint's* display solution, its user interface and standardized data storage capability. The Python API and the easy integration of external libraries, such as OpenCV, results in slim and readable code, which can be easily adapted to the

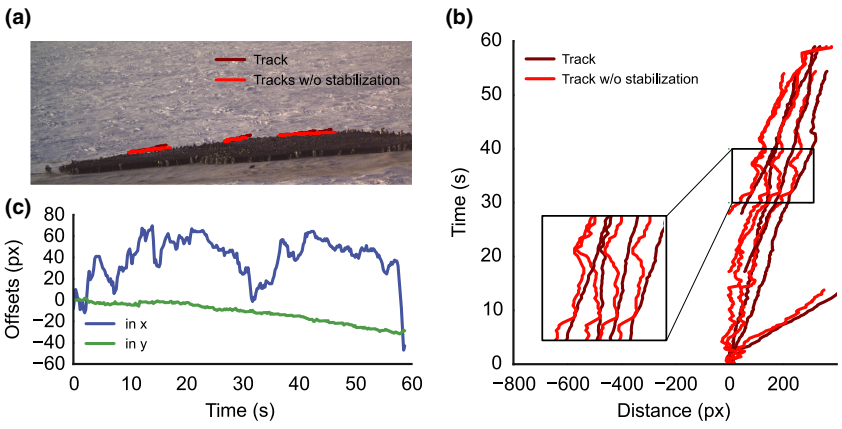


Fig. 6. Image stabilization for drift- and vibration-free trajectories. (a) Image of a video sequence with tracked penguins. (b) Trajectories with (dark red) and without (red) image stabilization. (c) Drift and vibration offsets in *x* and *y* direction.

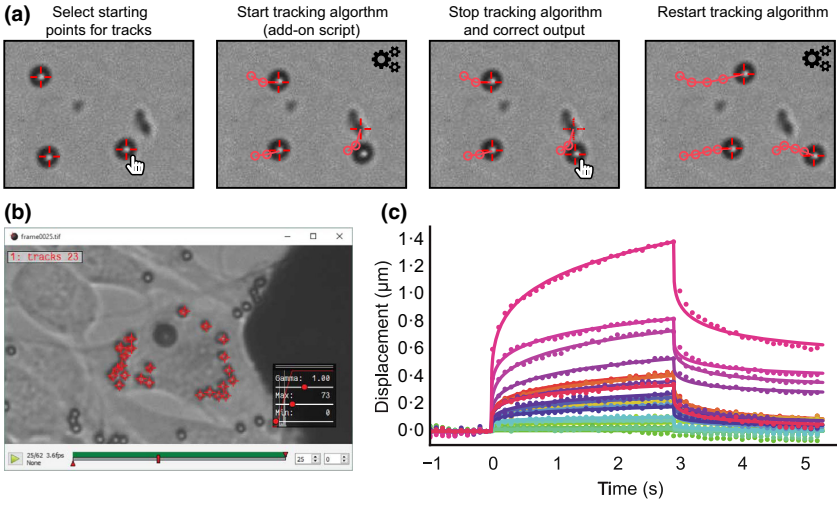


Fig. 7. Tracking of beads in microscope images. (a) Schematic of the tracking process: Starting positions are selected with *ClickPoints*, tracked, and displayed live in *ClickPoints*. If tracking errors occur, the user can correct the marker position in *ClickPoints* and restart the tracking algorithm. (b) *ClickPoints* interface showing the bead tracks and the image. (c) Bead displacement curves show the deformation of the cell during force application (0 s to 3 s) and during relaxation (>3 s).

task at hand. New applications can profit from the large and growing Python community (Shein 2015; Unpingco 2016) that offers an abundance of existing algorithms and packages, which can be combined with *ClickPoints*.

The example data set and evaluation code can be found as example "BeadTracking" at: http://clickpoints.readthedocs.io/en/latest/example_tweezer.html

DISPLAY OF RESULTS FROM EXTERNAL PROGRAMS

In this example, we use *ClickPoints* as a tool for display and analysis of results from a C++ bird tracking software.

Images of a 360° infra-red camera system (Zitterbart *et al.* 2013) are analysed with a real-time bird tracking software written in C++. The software uses the *ClickPoints* data base scheme to store the trajectories of the detected birds. Therefore, *ClickPoints* can be used to display the tracking results (Fig. 7a red), utilizing the data base to superimpose the results on the original footage. The frame by frame display is a valuable tool to visualize intermediate results, to gain further insight and intuition of an algorithm's workflow, e.g. to understand when and why the tracking algorithm fails.

Furthermore, *ClickPoints* helps to manually or semi-automatically generate ground truth labels for tracks (Fig. 8a and b). Such ground truth labels are required for an evaluation of tracking performance, e.g. to quantify accuracy or missed detections. The comparison of automatically generated tracks and manually labelled ground truth tracks is fast and convenient, as both exist in the same format and can be efficiently accessed by the *ClickPoints* API (Fig. 8c and d).

Even though the C++ tracking software runs completely independent of *ClickPoints*, it can take advantage of the data visualization and user interaction interface simply by storing its data in the *ClickPoints* format.

Software availability

ClickPoints is distributed under the GPLv3 license. It is used by research groups from diverse fields and is continuously improved and extended.

We provide a Windows installer as well as the raw zipped Python code for users who prefer to use their own Python installation. Instructions on how the software can be installed and used can be found in our documentation at: http://

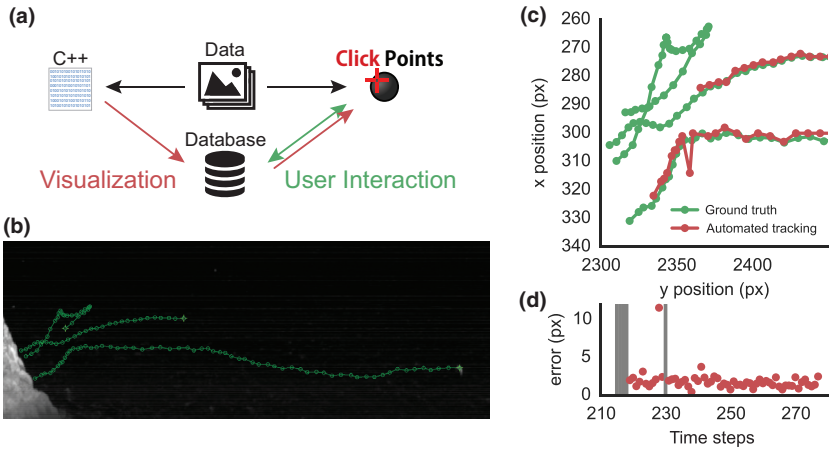


Fig. 8. Visualization and evaluation of Tracking results. (a) Diagram for interfacing external programs with *ClickPoints* for visualization and user interaction. (b) Manually labelled ground truth tracks (c) Overlay of automatic tracking results and ground truth tracks (d) Visualization of tracking error and missed detections.

clickpoints.readthedocs.io. The documentation provides detailed information on the use of the software and its various features and add-ons. In addition, we provide various examples to demonstrate features and applications of *ClickPoints*.

Conclusion and future directions

In this paper, we presented *ClickPoints*, an open-source software developed to combine and streamline the three main steps of image analysis - visualization, annotation and evaluation - in one program. *ClickPoints* allows the user to easily and efficiently review large data sets, mark interesting findings, and extract information. *ClickPoints* supports custom add-ons written in Python for adapting or extending the available tools. It provides an extensive API to efficiently store and retrieve results in SQL data base format. Utilizing *ClickPoints*' display, interface and storage functionality greatly reduces development times.

We believe that *ClickPoints* is a valuable tool for many scientific applications, and that it provides an efficient, flexible and affordable software solution for image time-series evaluation. We expect that *ClickPoints* will benefit from a growing community and their contributions.

Authors' contribution

R.C.G. and S.R. developed the software, R.C.G., S.R., B.F. and D.P.Z. wrote the manuscript.

Acknowledgements

We thank our collaboration partners for kindly providing example data sets from ongoing projects: Nadja Gerlitz for plant root images, Claus Metzner for the infrared bird tracking, and Céline Le Bohec, Anna Nesterova and Francesco Bonadonna for the Adélie and King Penguin recordings. This study was funded by the European Research Council Starting Grant MINATRAN 211166 and the Deutsche Forschungsgemeinschaft (DFG) grant FA336/5-1 in the framework of the priority program "Antarctic research with comparative investigations in Arctic ice areas".

Data accessibility

We do not archive data because this manuscript does not use data.

References

Ballarotti, M.G., Saba, M.M.F. & Pinto, J. (2005) High-speed camera observations of negative ground flashes on a millisecond-scale. *Geophysical Research Letters*, **32**, 1–4.

Bonakdar, N., Gerum, R., Kuhn, M., Spörrer, M., Lippert, A., Schneider, W., Aifantis, K.E. & Fabry, B. (2016) Mechanical plasticity of cells. *Nature Materials*, **15**, 1090–1094.

Bouguet, J.-Y. (2001) Pyramidal implementation of the affine lucas kanade feature tracker—description of the algorithm. Intel Corporation, Microprocessor Research Labs. http://robots.stanford.edu/cs223b04/algo_affine_tracking.pdf.

Bradski, G. (2000) The OpenCV library. *Dr Dobbs Journal of Software Tools*, **25**, 120–125.

de Chaumont, F., Dallongeville, S., Chenouard, N. *et al.* (2012) Icy: an open bioimage informatics platform for extended reproducible research. *Nature Methods*, **9**, 690–696.

Dhillon, V.S., Marsh, T.R., Stevenson, M.J. *et al.* (2007) ULTRACAM: an ultrafast, triple-beam CCD camera for high-speed astrophysics. *Monthly Notices of the Royal Astronomical Society*, **378**, 825–840.

Egea, I., Bian, W., Barsan, C., Jauneau, A., Pech, J.C., Latché, A., Li, Z. & Chervin, C. (2011) Chloroplast to chromoplast transition in tomato fruit: spectral confocal microscopy analyses of carotenoids and chlorophylls in isolated plastids and time-lapse recording on intact live tissue. *Annals of Botany*, **108**, 291–297.

Fossum, E. & Hondongwa, D.B. (2014) A review of the pinned photodiode for CCD and CMOS image sensors. *IEEE Journal of the Electron Devices Society*, **2**, 33–43.

Gerlitz, N. (2016) *Untersuchungen zur Funktion von Plasmodesmata in der Wurzelspitze von Arabidopsis thaliana*. PhD thesis, Friedrich-Alexander Universität, Erlangen-Nürnberg, Germany.

Hunter, J.D. (2007) Matplotlib: a 2D graphics environment. *Computing in Science and Engineering*, **9**, 99–104.

Janesick, J.R. (2001) *Scientific Charge-Coupled Devices*. SPIE Press, Bellingham, Washington, WA, USA.

Kankaanpää, P., Paavolainen, L., Tiitta, S., Karjalainen, M., Päivärinne, J., Nieminen, J., Marjomäki, V., Heino, J. & White, D.J. (2012) BioImageXD: an open, general-purpose and high-throughput image-processing platform. *Nature Methods*, **9**, 683–689.

Klein, A. (2014). ImageIO. https://imageio.github.io/

Kollmannsberger, P. & Fabry, B. (2011) Linear and nonlinear rheology of living cells. *Annual Review of Materials Research*, **41**, 75–97.

Kryder, M.H. & Kim, C.S. (2009) After hard drives-what comes next? *IEEE Transactions on Magnetics*, **45**, 3406–3413.

Le Bohec, C. (2013) Programme 137 of the Institut Polaire Français Paul-Emile Victor (PI: Céline Le Bohec). CNRS-Université de Strasbourg & Centre Scientifique de Monaco.

Reif, V. & Tornberg, R. (2006) Using time-lapse digital video recording for a nesting study of birds of prey. *European Journal of Wildlife Research*, **52**, 251–258.

Richter, S. (2011) *Entwicklung einer autonomen Beobachtungsplattform zum Einsatz in polaren Regionen*. Diploma thesis, Friedrich-Alexander Universität, Erlangen-Nürnberg, Germany.

van Rossum, G. (1995) *Python Tutorial*. CWI, Center for Mathematics and Computer Science, Amsterdam, The Netherlands.

Schindelin, J., Arganda-Carreras, I., Frise, E. *et al.* (2012) Fiji: an open-source platform for biological-image analysis. *Nature Methods*, **9**, 676–682.

Schneider, C.A., Rasband, W.S. & Eliceiri, K.W. (2012) NIH Image to ImageJ: 25 years of image analysis. *Nature Methods*, **9**, 671–675.

Shein, E. (2015) Python for beginners. *Communications of the ACM*, **58**, 19–21.

Skiljan, I. (2016) IrfanView. http://www.irfanview.de/

The Qt Company. (2016) Qt. http://www.qt.io/

Unpingco, J. (2016) *Getting Started with Scientific Python*. Springer International Publishing, Cham, Switzerland.

VideoLan Organization. (2016) VLC. http://www.videolan.org/vlc/

Zitterbart, D.P., Kindermann, L., Burkhardt, E. & Boebel, O. (2013) Automatic round-the-clock detection of whales for mitigation from underwater noise impacts. *PLoS ONE*, **8**, e71217.

Received 26 August 2016; accepted 21 October 2016
Handling Editor: Timothée Poisot