EURASIP Journal on Image and Video Processing
a SpringerOpen Journal

RESEARCH Open Access

A two-fly tracker that solves occlusions by dynamic programming: computational analysis of *Drosophila* courtship behaviour

Christian Schusterreiter[1,2*] and Wilfried Grossmann[3]

Abstract

This paper introduces a two-fly tracker which focuses on an approach to model and to solve occlusions as an optimization problem. Automated tracking of genetic model organisms is gaining importance since geneticists and neuroscientists have biological tools to systematically study the connection between genes, neurons and behaviour by performing large-scale behavioural experiments. This paper is about a fly tracker that provides automated quantification for such functional behaviour studies on *Drosophila* courtship behaviour. It enables measurement and visualization of behavioural differences in genetically modified fly pairs. The developed system provides solutions for all major challenges that were identified: arena detection, segmentation, quality control, resolving occlusions, resolving heading and detection of behaviour events. Among all challenges especially resolving occlusions turned out to be of particular importance and huge effort was invested to resolve that particular problem. Our tests show that our system is capable to identify flies through an entire video with an accuracy of 99.97%. This result is achieved by combining different types of local methods and modeling the global identity assignment as an optimization problem.

Keywords: Fly tracker; Ethogram; Occlusion; Dynamic programming; *Drosophila*; Courtship behaviour; Quantification; Pattern recognition

1 Introduction

1.1 Motivation and goals

A fundamental question in neuroscience is to understand the relation between genes, brains and behaviour: Genes encode hard-wired neuronal circuits in the nervous system. For innate behaviours - like reproductive behaviour of insects - such neuronal circuits produce observable stereotypic motor outputs.

The fruit fly *Drosophila melanogaster* has a set of innate behaviours that are hard-wired in the nervous system. Several innate behaviours of *D. melanogaster* are sex-specific. In combination with the availability of genetic and molecular tools, the fruit fly is a common model organism to study how the nervous system generates behaviours.

Drosophila courtship is a robust and sex-specific behaviour that has been characterized through multiple genetic screens. Many genes that regulate male and female courtship behaviour have already been identified. It was a big surprise that a complex behaviour like courtship is regulated by a few sets of genes [1,2], and it is strongly believed that these genes interact with cascades of downstream genes that regulate individual parts of the behaviour.

Currently geneticists and neuroscientists perform large-scale experiments in order to systematically identify genes and neurons that are involved in specific steps of courtship behaviour. Quantification of these experiments and classification of different behaviours turned out to be a very time consuming and tedious task; thus, it was the major bottle neck of large-scale behaviour screens for a long time.

Automated tools aim to support such large-scale experiments. Saving time is one important factor, but in addition, automation limits human error and extends possibilities for robust, objective and reproducible analysis.

*Correspondence: christian.schusterreiter@gmail.com
[1] I.M.P. Research Institute of Molecular Pathology, Dr. Bohr-Gasse 7, 1030 Vienna, Austria
[2] Present address: University of Oxford, Department for Computer Science, Wolfson Building, Parks Road, Oxford OX1 3QD, UK
Full list of author information is available at the end of the article

This paper describes the development of a system, which translates courtship behaviour videos into formal descriptors using computer vision and statistical methods. The descriptors allow ethogram-like descriptions of complex courtship behaviour patterns for each fly. A special feature of our system is the identification of individual flies through the entire video by solving the occlusion problem with very high accuracy.

1.2 Related work

When the project was initiated, only a few trackers [3-6] existed for a different model organism called *Caenorhabditis elegans*. These trackers mainly analyzed the worm's movements and quantified turn direction versus straight movement. They excluded frames where worms occluded each other. The only published fly tracker [7] analyzed the fly locomotion behaviour.

In 2008 Perona published an automated fly tracker for courtship and aggressive behaviour [8,9] and initiated a transition from manual to automated scoring. Simultaneously, Schusterreiter developed a tracker [10] that measures the courtship index and captures courtship sub-behaviours.

In particular, the work of Dankert et al. [8] has similar aspects to this paper as it also introduces a two-fly tracker and tackles similar challenges. Our tracker mainly differs in three aspects: It was initially designed to process unseen videos that are not specifically recorded for automated tracking and therefore comes with an arsenal of quality boosting and quality control methods. Second, we spend a huge effort to tackle the occlusion problem, which was probably less critical for the application scenarios of [8]. Finally, our system offers top-down and bottom-up classifiers for courtship behaviour, while the other system offers top-down classifiers only but for both courtship and aggressive behaviours.

Branson et al. [9] quantifies behaviour of multiple flies, while our system is specifically optimized for two flies per chamber. Our system deals with flies turning their head up in z-direction and flies occluding each other in z-direction by improved software analysis, while the tracker [9] attacks these problems by improvements of the recording setup [11] that significantly decrease difficult occlusion cases.

Hoyer et al. [12] used a tracker that quantified aggression behaviour by a user-defined lunge counter and required one of the two male flies to be painted with a white dot on the back. Similarly, the identity tracking method introduced in [13] enabled biologists to genetically mark flies by a camera-detectable fluorescence marker. In contrary, our system is in principle capable to incorporate detectable color differences but does not *require* to mark flies.

The work introduced within this paper was developed independently from related work; however, some user-defined classifiers of the postprocessor were defined after the classifiers in [8] have been studied. Further similarities, like choosing the Hungarian algorithm for identity assignment in unoccluded sequences or circular arena detection by the Hough transform, are coincidental.

This paper introduces a two-fly tracker and focuses mainly on resolving occlusions as an optimization problem. It is organized as follows: Section 2 introduces the main components of the entire system. Section 3 starts with the basic definitions for the occlusion problem (Section 3.1) and local methods for occlusion assignments (Section 3.2). The solution of the occlusion problem as an optimization problem is stated in Section 3.3, and a dynamic programming algorithm that solves this optimization problem is presented in Section 3.4. Results of the approach will be discussed in Section 3.5. Section 4 shows that the same algorithm is capable to solve the heading problem. Finally, Section 5 provides a summary, discusses properties and results of the optimization algorithm and outlines future work.

2 A two-fly tracker

In general, automated tracking is a data *densification* process that takes high amounts of *video data* having low information content and turns them into low amounts of *relevant features* having high information content. Our system comes with two major steps: an *image processing step* where raw video data is transformed into a time series representation and a *pattern recognition step* where biologically relevant events are detected within that time series. The image processing part further subsumes several data transformation and data cleaning steps while the data is still in its image representation. It thus boosts quality and plausibility of image data before the time series is extracted and ensures that minimum quality standards are met. In case videos are detected to be inappropriate for downstream computation steps, they are rejected as early as possible in order to save computation time.

The system architecture consists of several *modules* named *preprocessor*, *tracker*, *postprocessor* and *annotationTool*. The preprocessor and the tracker cover the image processing part, the postprocessor derives advanced attributes and covers the pattern recognition part. The workflow between modules is straightforward: information flows from the preprocessor through the tracker to the postprocessor module. The only two-way interacting component is the annotationTool; it interoperates with postprocessed data (*cf.* Figures 1 and 2).

The following paragraphs contain brief descriptions for each module; more details for main functionality may be found in [10].

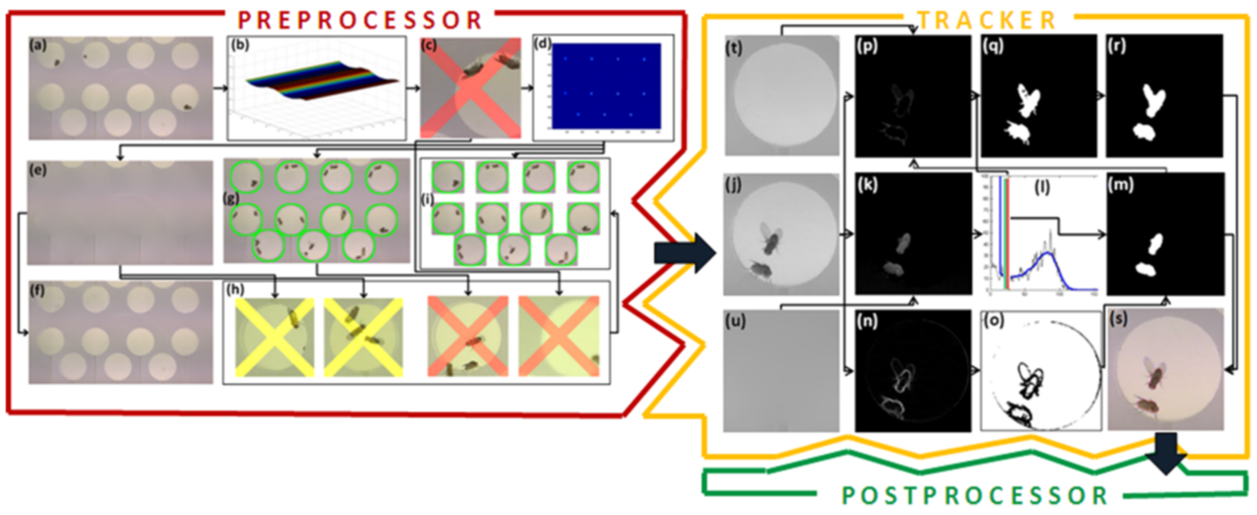


Figure 1 Schematic overview of the image processing part. *Preprocessor (red shape):* **(a)** extract median background; **(b)** illumination correction curve; **(c)** chamber rejected in movement detection step; **(d)** accumulated peaks for circular arena detection; **(e)** rigorously smoothed background for body segmentation; **(f)** cautiously smoothed background for wing segmentation; **(g)** watched boundary for each detected arena; **(h)** chambers rejected by quality control, witness pictures capture reason for rejection; and **(i)** individual chambers after arena splitting step. *Tracker (yellow shape):* **(j)** gray values of an original frame of a single chamber video; **(k)** subtracting rigorously smoothed background for body region segmentation; **(l)** threshold determination: body region threshold (red line) and wing region threshold (green line) are determined within the gray value histogram (black line), respectively, a smoothed histogram (blue line); **(m)** binarized fly body region **(n)** gradient magnitude values, bright values indicate edges; **(o)** mask region for morphological reconstruction; **(p)** subtracting cautiously smoothed background for wing region segmentation, body and non-arena regions are black; **(q)** binarized wing region with legs; **(r)** fly wing region without legs and with filled holes; **(s)** original image with segmentation results, body region in red, wing region in blue; **(t)** cautiously smoothed background; and **(u)** rigorously smoothed background.

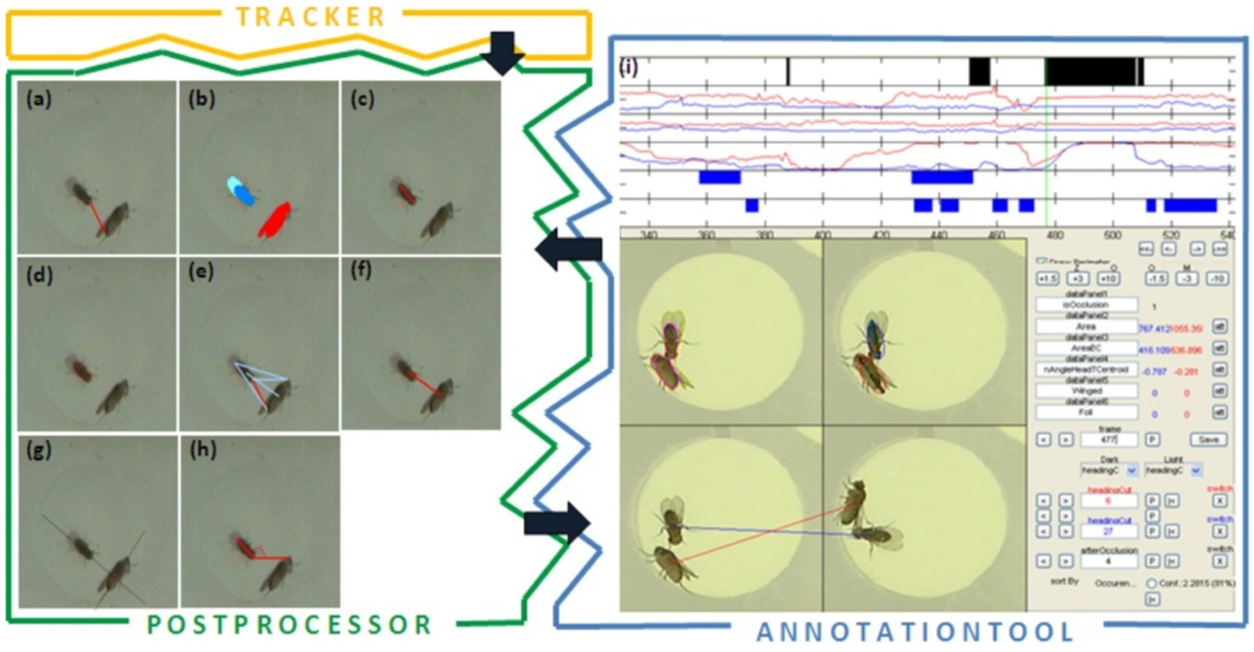


Figure 2 Schematic overview of Postprocessor and AnnotationTool. *Postprocessor (green shape):* **(a)** *distHeadTail*, distance between flies *Head* and other flies *Tail*; **(b)** *nArea* and *nwArea*, size of body area respective wing area; **(c)** *flyLen*, *MajorAxislength* of fitting ellipse and distance between Head and Tail; **(d)** *Eccentricity*, measure for roundness depending on relation between MajorAxisLength and *MinorAxisLength* of covering ellipses; **(e)** *minDist*, shortest of five depicted distances; **(f)** *distance*, distance between the flies centers-of-gravity; **(g)** *hrndirectednessR5*, combined measure of orientation angle and distance; and **(h)** *nAngleHeadHead*, angle between fly and other flies' Head. **(i)** Screenshot of *AnnotationTool (blue shape)*. Data panels for attribute visualizations on top, machine-annotated video panels, below.

The preprocessor identifies individual arenas (*cf.* Figure 1a) and boosts video quality for each frame. Quality improvement encompasses illumination correction based on an illumination correction curve (*cf.* Figure 1b) and elimination of arenas where camera movement, intruding objects or not exactly two flies were detected (*cf.* Figure 1c,h,i). In this process, approximately 2% to 5% of chamber videos are rejected. For arena detection and all further image processing two gray level pictures are essential: the so-called rigorously smoothed background (Figure 1e) and the cautiously smoothed background (Figure 1f). Arenas are detected by a circular Hough transform which identifies number, position and diameter of the arenas (Figure 1d). Arena boundaries (Figure 1g) are watched by an intrusion detector. Finally, videos are split into individual arenas (Figure 1i), each handed separately to the tracker.

The tracker shown in Figure 1j,k,l,m,n,o,p,q,r,s takes single arena videos and corresponding smoothened backgrounds as input (*cf.* first column of the tracker figure, Figure 1j,u,t). The second column shows the arena after subtracting the smoothened backgrounds (first and second rows, Figure 1p,k) and the results of a gradient procedure in the third row (Figure 1n). Further processing is based on the gray level histogram shown in the middle of the third column (Figure 1l). Two different thresholds are used for body and wing detection. Using the thresholds for body region together with the gradient pictures for the boundaries, we obtain a picture for the body region (Figure 1m). Using this result in connection with the threshold for wing detection, we obtain from (Figure 1p) the first image of the binarized body-wing region (Figure 1q). This image is further improved by filling holes (Figure 1r). Resulting body regions (Figure 1m) and wing regions (Figure 1r) are marked in the original image in Figure 1s. Extraction of *primary* attributes directly from images (Figure 1m) and (Figure 1r) concludes the image processing step. These attributes build the interface to the postprocessor module. We derive for both body and wing region: the number of pixels *Area*, the region *Perimeter* and the center of gravity *Centroid*, *Orientation*, *MinorAxisLength* and *MajorAxisLength* of a covering ellipse.

The tracking process is accompanied by number of quality control steps like checking for intrusions into the watched boundary around a chamber (*cf.* Figure 1i) and evaluation of tracked primary attributes' plausibility.

The postprocessor covers the pattern recognition part of the system and searches for biologically relevant events. As a first step tracking data is *normalized* such that all attributes are comparable across different videos. From normalized primary attributes we compute secondary attributes that allow definition of behavioural patterns like following, wing extension or copulation. Figure 2 shows some of these attributes. They capture specific fly constellations (Figure 2a,e,f,g,h) or shapes (Figure 2b,c,d). Transformation of these attributes into behaviour patterns requires identification of individual flies in each video frame and detection of each fly's head and tail. Fly identification is rather simple as long as the tracker distinguishes separate regions for each fly in so-called unoccluded frames. For sequences of unoccluded frames, fly identities are carried through successive frames by solving an assignment problem for position characteristics with the Hungarian algorithm [14].

In case of occluded frames, the frames where fly bodies overlap (occlude) each other, primary and secondary attributes are computed after solving the occlusion problem. A solution assigns a matching for fly identities before and after each occlusion (see Section 3). According to these matchings, occluded primary attributes are approximated by interpolations. Having primary attributes for every single frame, the postprocessor then determines the head and tail for each fly body (see Section 4) and computes all other secondary attributes.

The system may then apply machine-learned or user-defined classifiers to detect relevant behaviour events. Detected events are protocolled in color-coded *ethograms* (*cf.* Section 5.2) and excel sheets.

The annotationTool interacts with the postprocessed data. It supports attribute inspections and overruling of machine decisions. Manually tweaked postprocessing data is re-postprocessed to ensure consistent data views and to avoid time-consuming recalculations during online annotations.

The screenshot in Figure 2i contains data panels on top that visualize attributes for both flies, video panels that depict video frames with tracked perimeter, automatically annotated heading and interpolated ellipses and an occlusion panel that visualizes fly identification across an occlusion. Control panels on the right are for video navigation, attribute selection and manual annotation.

We further implemented a *webinterface* to bulk-submit processing tasks to a computer cluster and to manage videos and tracking results.

3 Resolving occlusions

3.1 Problem definition

When examining social interactions, the aim is to capture behaviour especially when the individuals are close to each other. Therefore, it is necessary to identify individual flies throughout the entire video even if they overlap or occlude each other. If the two flies move close together and their body regions overlap such that the segmentation method detects only a single body region for both flies, then assigning fly identities becomes a difficult task for a computer. Even for humans, it is sometimes difficult or even impossible to allocate individuals correctly after two flies have overlapped completely.

Since it is essential to be able to allocate the individual flies for the behavioural studies, the occlusion problem was a key challenge in system development and huge effort was invested to tackle the occlusion problem.

Figure 3 shows four cases of occlusion to further illustrate the problem. The majority of occlusion cases are very similar to the ones depicted in Figure 3a,b; the one in Figure 3a depicts a social interaction that typically happens in occluded scenes. Our system reliably resolves such occlusions (and also detects the wing extension during occlusion in Figure 3a). A rare case where our system is wrong (Figure 3d) is further discussed in Section 3.5.

For resolving occlusions, the sequence of all video frames V is partitioned into alternating σ and τ sequences. While σ sequences contain unoccluded frames where both fly bodies are detected separately, τ sequences contain occluded frames where the two fly bodies are merged into one larger region or where no flies at all were detected.

Formally, σ and τ sequences are defined as follows:

Definition 1. Let f be a frame. Function $o(f)$ is defined as $o(f) = 1$ in case f is occluded and $o(f) = 0$ otherwise. Let f_0 refer to an empty frame. $o(f_0) = 1$.

Definition 2. Let V be a sequence of frames $f_i, f_i \in V$. A sequence $\sigma \subseteq V$ contains a set of successive frames f_i with $\forall f_i \in \sigma : o(f_i) = 0$; a sequence $\tau \subseteq V$ contains successive frames with $\forall f_i \in \tau : o(f_i) = 1$.

Corollary 1. *The border for a partitioning Φ of V that consists of alternating σ and τ sequences is marked by $\Delta[o(f_i)]$. The partitioning Φ_0 of $V_0 = f_0 \cup V \cup f_0$ is guaranteed to start and end with a τ sequence.*

Definition 3. The occlusion problem is finding the best overall assignment of fly identities in all unoccluded sequences using observable fly attributes.

The problem is solved in two steps. First, we calculate *local scores* for the possible assignments of the fly identities in a subsequence $(\sigma_i, \tau_i, \sigma_{i+1})$, using only information in the occluded sequence τ_i and its two enclosing unoccluded sequences σ_i and σ_{i+1}. These scores can be interpreted as probabilities for a matching and are determined by local methods called *t-methods*. Different local methods are introduced in Section 3.2. The occlusion sequences in Figure 3a,b depicts rather trivial occlusion cases where most local methods are successful. The sequences in Figure 3c,d shows cases that may deliver diverged results from different local methods.

Resolution of such ambiguities is done in the second step by (a) formalizing the occlusion problem as a global optimization problem in subsection 3.3 and (b) solving it with a dynamic programming approach (see subsection 3.4).

3.2 Local methods

Local methods or t-methods are associated with a τ sequence and aim to provide an assignment for the identifiers of its enclosing σ sequences. Each t-method

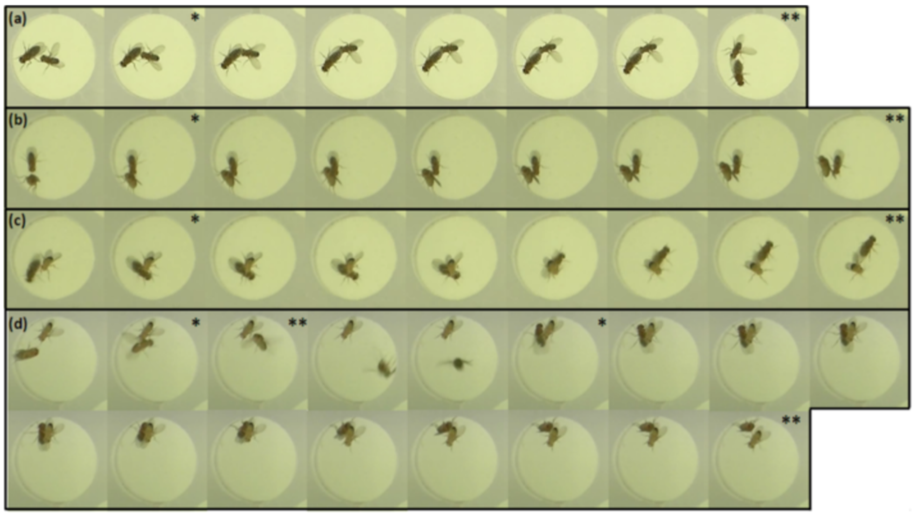


Figure 3 Sample occlusion sequences of increasing difficulty. Single asterisk marks first occluded frame, double asterisk the first frame after an occlusion. **(a)** A trivial case that most methods should get right. **(b)** Sequence that some size-based methods may get wrong as the typically larger female fly turns up and appears to be smaller. **(c)** A sequence that most point-based methods may get wrong. **(d)** Difficult sequence constellation containing a fly jumping into an occlusion, visualizes last error instance discussed in Section 3.5.

computes a *t value*, respectively, a *t score* for each assignment that resembles its certainty.

In general, t-methods may be differentiated into *attribute-based methods* following a *merge-and-split* approach, *point-based methods* following a *straight-through* approach [15] and *combination methods*.

3.2.1 Attribute-based methods

Follow a classical merge-and-split approach. The idea of attribute-based methods is pretty simple: compare the values of known fly attributes before and after the occlusion and assign pre-occlusion flies to best matching post-occlusion flies. The particular set of characteristic attributes that are used to re-identify flies may vary.

Although any attribute can be taken into account, we will first focus on size-based attributes. An initial motivation for size-based methods was the known size difference between male and female flies.

In general, attribute-based methods match flies according to the mean, maximum, minimum and any other aggregation of sizes before occlusion and after occlusion.

Method *siz*1 aggregates an eccentricity-corrected size attribute $\text{AreaEC} = \text{Area}\sqrt{1 - \text{EccentricityC}^2}$, $\text{EccentricityC} = \frac{\text{Eccentricity}}{1+e^{5\cdot\text{Eccentricity}}}$ from whole σ sequences and compares probabilities that indicate which fly is bigger. Figure 2i (blue shape, second and third data panel) visualizes the attributes Area and AreaEC next to each other (particularly note frames 460 to 480 when the larger fly turns up in z-direction). When using attribute AreaEC method siz1 solves all occlusion cases in Figure 3, while straight incorporation of Area would get sequence Figure 3c wrong.

Method posm compares Centroids from the last frame before the occlusion to the first frame after the occlusion and computes a score $\nu \in (-1, +1)$. The score aggregates Centroid distances that indicate a matching of fly 1 before the occlusion being fly 1 after the occlusion, $o_b^1 \rightharpoonup o_a^1$ (and correspondingly $o_b^2 \rightharpoondown o_a^2$) versus the opposite matching. Normalized by all involved distances, the score $\nu = -\frac{(o_b^1 \rightharpoonup o_a^1)+(o_b^2 \rightharpoondown o_a^2)-(o_b^1 \rightharpoonup o_a^2)-(o_b^2 \rightharpoondown o_a^1).}{(o_b^1 \rightharpoonup o_a^1)+(o_b^2 \rightharpoondown o_a^2)+(o_b^1 \rightharpoonup o_a^2)+(o_b^2 \rightharpoondown o_a^1).}$ is guaranteed to be between -1 and $+1$ and is negated to prefer short distances. Result scores ν indicate an identity assignment by $\text{sign}(\nu)$ and the method's certainty about their assignment by $|\nu|$.

3.2.2 Point-based methods

follow the straight-through approach where a point set $\mathfrak{C}$ that is traced 'straight through' the occlusion states. The object's perimeter turned out to be a good choice for $\mathfrak{C}$, it outperformed all other tested point set candidates by solution quality or computation time.

The aim of point-based methods is to assign identifiers from the state before the occlusion b, the last frame where both flies have been identified, to the state after the occlusion a, the first frame where both flies are identified again. For this reason, point sets $\mathfrak{C}$ are extracted before and after the occlusion and each point is associated with an identifier of the two separately detected flies. Then, for each frame during the occlusion, the point set is extracted and associated identifiers are carried over from its predecessor frame by a nearest neighbour assignment using Voronoi diagrams [16]. At the end, the identifier set carried through the occlusion $\hat{\mathfrak{C}}_a$ is compared with the freshly partitioned point set $\hat{\mathfrak{C}}_{a'}$ and a score ν is derived that resembles how associated identifiers of the characteristics in $\hat{\mathfrak{C}}_a$ and $\hat{\mathfrak{C}}_{a'}$ match. The score particularly aggregates the sum of identifier votes from $\mathfrak{C}_a$ that indicate mapping identifiers o_b^1 to o_a^1 and o_b^2 to o_a^2 minus the votes for mapping o_b^1 to o_a^2 and o_b^2 to o_a^1, normalized by the sum of all votes, $\nu = \frac{(o_b^1 \rightharpoonup o_a^1)+(o_b^2 \rightharpoondown o_a^2)-(o_b^1 \rightharpoonup o_a^2)-(o_b^2 \rightharpoondown o_a^1).}{(o_b^1 \rightharpoonup o_a^1)+(o_b^2 \rightharpoondown o_a^2)+(o_b^1 \rightharpoonup o_a^2)+(o_b^2 \rightharpoondown o_a^1).}, \nu \in (-1, +1)$. Resulting scores ν again indicate a suggested identity assignment and the certainty about this assignment result in $\text{sign}(\nu)$ respectively $|\nu|$.

The major weakness of all point-based methods comes with the nearest neighbour assignment. Due to the fact that each pixel takes over the identifier of its nearest pixel in the previous frame, crossing flies are likely to be mis-scored. In fact, all mis-scores and 'don't know' cases that scored with a value of 0 result from this known issue. The latter case especially comes up when the occluded region moves over longer distances. A method variant bocT therefore aims to compensate such movements by applying rigid transformation between successive frames and reduces the effect of that particular weakness.

The boc method and its variants turned out to be particularly reliable for occlusion cases like the ones in Figure 3a,b and are likely to get cases like the one in Figure 3c wrong. Although point-based methods have known difficulties when dealing with crossing flies - they still correctly solve between 90% and 95% of our test case set (see Section 3.5) and typically give low certainty values when they are wrong.

3.2.3 Combining meta-methods

Aim to boost scores from individual local methods by machine learning techniques. For this reason, we implemented a large number of attribute-based, point-based and other methods; we extracted observable attributes from occluded blobs, in particular, the duration of the occlusion and its minimum number of pixels (providing information about a 'maximal degree of occlusion') turned out to provide good occlusion characterizations. After computation of all decision and score results from all implemented methods, a *meta-method* was trained by standard machine learning approaches. The Classification and Regression Trees (CART) turned out to be a useful approach; although alternative meta-method approaches performed equally well, the tree approach was chosen because of its intuitive and easy understandable rule-based decisions.

The experiment results in Section 3.5 contain results from a cross-validated CART method where local methods, each having an accuracy of 90% to 95%, are bundled into a combined t-method with about 99% accuracy.

Alternatively, the probability-converted score of independent methods may be combined by the *Dempster combination* [17,18], which allows to mathematically combine evidences from different sources into a combined degree of belief. The Dempster combination is defined as follows:

Definition 4. Let e_1 and e_2 be two independent evidences. The Dempster combination of these two evidences is defined as $e_1 \otimes e_2 = \frac{e_1 \cdot e_2}{1-K}, K = (1-e_1) \cdot e_2 + e_1 \cdot (1-e_2)$.

The definition above allows to cumulatively combine evidences from multiple t-methods into new t-probabilities. The Dempster combination may also be used to combine independent s-methods (s-methods are introduced in Section 3.3 below).

3.3 Occlusions as global optimization problem

Local methods process cases linearly and assess occlusion sequences independently one after another. Therefore a wrong identity assignment is passed on through the entire video as identities are swapped from that wrong assignment on and therefore mis-assigned up to the end of the video (*cf.* in Section 3.3, first example).

In order to overcome this error propagation problem we complement t-methods with so-called *s-methods*. While t-methods are associated with τ sequences, s-methods are associated with unoccluded σ sequences where both flies are detected. These s-methods aim to discriminate and re-identify the two detected flies much like in the merge-and-split approaches introduced in Section 3.2. However, while merge-and-split t-methods assess and compare characteristics of the σ sequences directly before and after an occluded τ sequence, the characteristics for s-methods require to be *comparable during the whole video*. Similar to the t-methods, a s-method provides a *s-score* for each assignment that resembles its certainty.

The comparability of s-methods is a key property to overcome the error propagation problem that comes when using t-methods only and is essential for the optimization approach described in this section. The following two examples underline the difference between s- and t-methods:

The size-based method siz1 (see Section 3.2) aims to match flies before an occlusion (in sequence σ_b) to flies after an occlusion (in sequence σ_a) according to an observed size difference. The bigger fly is assigned to the bigger fly and the smaller fly is assigned to the smaller fly. Such a size-based method may easily be generalized to become a s-method, since the discriminating characteristic - the fly size - is comparable during the whole video. In other words, flies in an arbitrary unoccluded sequence σ_k may be matched to flies of every other unoccluded sequence σ_i, such that the bigger flies are assigned to each other.

On the contrary, the position-based method posm (see again Section 3.2), which aims to match flies according to their position, is *not* suitable for a s-method generalization. Obviously, longer time spans between two sequences σ_k and σ_i will lead to improper results.

In general anatomical features, e.g. size or eye color, suggest suitable s-methods implementations. In principle any measurable anatomical or otherwise constant feature (like a painted mark) that discriminates the flies is applicable.

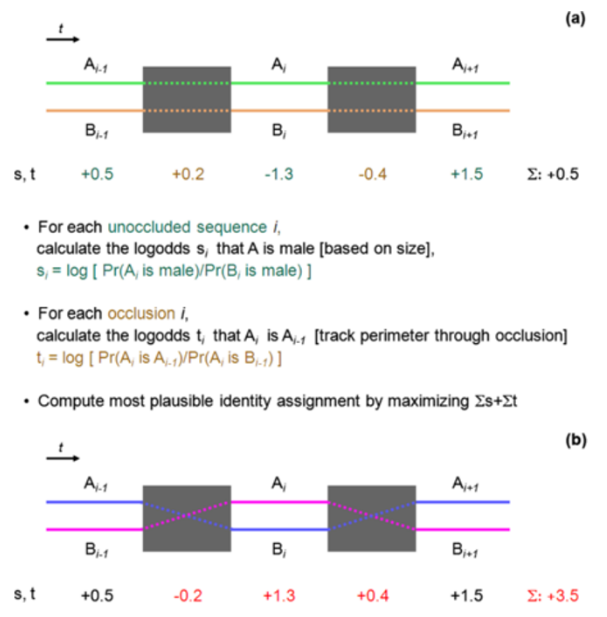
An intuitive combination for s- and t-methods would be to select scores where a s-method is absolutely sure and to then treat corresponding identity assignments as 'fix points.' Then t-methods may be used for low-score cases between these fix points only. Such an approach would limit the intrinsic problem coming with t-methods as mis-assignments would only be propagated up to the next fix point. The introduced optimization approach is a generalization of this idea and enables that s-methods and t-methods *correct each other*.

In order to ensure comparability of method results, their scores are converted to *probabilities* as probability values are *comparable* and *combinable with each other*.

In theory the conversion is done by empirically determining the distribution of score values per method and then deriving a value p from a score and the method's specifically given score distribution. In practice, using a linear approximation turned out to lead to sufficiently accurate results for all incorporated methods (see Section 3.5).

Finally, s values and t values are defined for each methods as *logodds* which are derived from these (approximated) probability values, $\nu = \ln(\frac{p}{1-p})$. Logodds inherit all comparability and combinability properties and further provide two desirable mathematical properties: (1) logodds of counter probabilities correspond to an inversion in sign, $\nu' = \ln(\frac{1-p}{p}) = -\nu$ and (2) logodds are combinable by addition.

Figure 4a explains the values assignment to σ and τ sequences by an example: After the video is split into alternating σ and τ sequences (occluded τ sequences are marked by gray boxes in Figure 4), the s and t scores are computed. The system incorporates size-based method siz1m as s-method and the point-based method boc as t-method (see Section 3.2).



- For each unoccluded sequence i,
  calculate the logodds s_i that A is male [based on size],
  $s_i = \log [\ \Pr(A_i \text{ is male})/\Pr(B_i \text{ is male})\ ]$
- For each occlusion i,
  calculate the logodds t_i that A_i is A_{i-1} [track perimeter through occlusion]
  $t_i = \log [\ \Pr(A_i \text{ is } A_{i-1})/\Pr(A_i \text{ is } B_{i-1})\ ]$
- Compute most plausible identity assignment by maximizing $\Sigma s+\Sigma t$

Figure 4 Sample problem instance for the optimization approach. (a) Identified flies over time: A in green and B in orange. Occluded sequences are marked by gray boxes. *s* values associated with non-occluded sequences are in blue; *t* values for occluded sequences are in brown. The text summarizes the computation of *s* and *t* values and color-codes keywords and definition in blue and brown for *s*-respective *t* values. **(b)** Visualization of same instance as in **(a)** after a flip operation; fly identifiers are switched and switched back within the two gray boxes. Red values show the flipped *s* and *t* values and their updated sum.

For each unoccluded sequence, σ the two detected flies are arbitrarily named fly A and fly B and the s-method siz1m - a variant of the sign test that provides good approximations for short sample sizes - computes the probability p that A is the bigger fly. The logodds s are derived from p and assigned to each unoccluded sequence (see cyan values in Figure 4a). Positive values indicate that fly A is the bigger fly, negative values that it is assumed that B is the bigger fly.

For each *occluded* τ sequence, the probability that fly A *before* the occlusion remains fly A *after* the occlusion is derived from a t-method. Method boc carries identifier information through the occlusion (*cf.* point-based methods in Section 3.2 for a more detailed description). Again, the logodds are computed, and the resulting t values are assigned to the occluded sequences (see brown values in Figure 4a). While s values correspond to the probability that fly A is the smaller fly (in Figure 4 written as 'A is male'), t values correspond to the probability that fly A in the σ sequence *before* the occlusion corresponds to fly A in the σ sequence *after* the occlusion. The (potentially artificial) τ sequences at the beginning and the end of V_0 are assigned with t values of 0.

Having all s values and t values in place, the occlusion resolvement problem may now be treated as an optimization problem. The proposed optimization algorithm (Section 3.4) uses a *dynamic programming* approach to compute the most plausible identity assignment by maximizing $\sum s + \sum t$ under a *flip operation*.

A *flip operation* affects two occluded sequences τ_i and τ_j and all unoccluded sequences between them. But most importantly, it does *not* affect the identities in sequences *outside* these two occlusions. All sequences before τ_i and after τ_j remain unchanged.

Figure 4b depicts the flow of identifiers in Figure 4a after a flip operation between the two occlusions drawn as gray boxes. In the first occlusion, identities of the flies are swapped, which results in swapped identifiers in the sequence in the middle as well, and in the second occlusion, identities are swapped back, making 'flip' a *local* operation only.

Swapping identities mathematically corresponds to inverting the sign of s and t values.

In Figure 4 the flipped identities in Figure 4b have a total value $\sum s + \sum t = 3.5$ and are therefore more plausible than identities in Figure 4a with $\sum s + \sum t = 0.5$.

Formally, this flip operation is defined as follows:

Definition 5. Let $\mathcal{S}$ and $\mathcal{T}$ be sequences of s values and t values associated with sequences σ and τ, such that $s_i \in S$ denotes the s value for σ_i and $t_i \in T$ denotes the t value for τ_i. The operation flip(i,j) on $\mathcal{S}$ and $\mathcal{T}$, defined as function $(\mathcal{S},\mathcal{T})' = \text{flip}(i,j,\mathcal{S},\mathcal{T})$, reverts the signs of t_i and t_j and of all $s_k, i \leq k < j$ in between them.

This flip operation comes with a number of desirable mathematical properties. It is obviously commutative and associative.

Definition 6. Let flip(i,j) and flip(k,l) be flip operations on V. The combined operation of both flips is denoted as flip$(i,j) \cup$ flip(k,l).

Since *flip* is commutative, the order of resolving the underlying individual operations does not matter. Flip is obviously semi-idempotent flip$(i,j) \cup$ flip$(i,j) = \varnothing$ and therefore concatenable flip$(i,k) \cup$ *flip*$(k,j) =$ flip(i,j) since flip$(k,k) \cup$ flip$(k,k) = \varnothing$.

Lemma 1. *Let i,j,k,l be indices for $\mathcal{V}$ with $i \leq j \leq k \leq l$. Then flip$(i,k) \cup$ flip$(j,l) =$ flip$(i,j) \cup$ flip(k,l).*

Proof. flip$(i,k) \cup$ flip$(j,l) =$ (concatenable)
(flip$(i,j) \cup$ flip$(j,k)) \cup ($flip$(j,k) \cup$ flip$(k,l))$ = (associative)
flip$(i,j) \cup$ (flip$(j,k) \cup$ flip$(j,k)) \cup$ flip(k,l) = (semi-idempotent)
flip$(i,j) \cup$ flip(k,l) □

These properties of flip encourage the definition of a normal form $\mathcal{F}_\perp$ for a set flip-operations.

Definition 7. Let f be a flip operation $f = \text{flip}(i,j), f \in \mathcal{F}$, $|f|$ denote the number of sequences affected by flip operation flip(i,j) and $|\mathcal{F}|$ therefore be $|\mathcal{F}| = \sum_{f \in \mathcal{F}} |f|$. Further, let $\mathcal{V}' = \mathcal{F}(\mathcal{V})$ denote the result of the application all flip operations in $\mathcal{F}$, and $\mathfrak{F}$ be the infinite set of all flip operation sets that are equivalent to $\mathcal{F}$, $\mathfrak{F} = \{\mathcal{F}_i | \mathcal{F}_i(\mathcal{V}) = \mathcal{F}(\mathcal{V})\}$. The normal form $\mathcal{F}_\perp$ of $\mathcal{F}$ is defined as the set of flip operations flip(i,j) with $i < j$ that affects the smallest amount of sequences but still delivers the same result, $\forall \mathcal{F}_i, \mathcal{F}_\perp \in \mathfrak{F} : |\mathcal{F}_\perp| \leq |\mathcal{F}_i|$.

Corollary 2. *A normal form $\mathcal{F}_\perp$ of $\mathcal{F}$ does neither contain double-flip operations flip$(i,j) \in \mathcal{F}_\perp$, flip$(k,l) \in \mathcal{F}_\perp \longrightarrow (i,j) \neq (k,l)$ nor flip overlaps that would contain double-flip operations, flip$(i,j) \in \mathcal{F}_\perp$, flip$(k,l) \in \mathcal{F}_\perp, i < l \longrightarrow j < k$. The properties $i < j$, $k < l$ and transitively $i < k$ and $j < l$ follow from the convention that $i < j$ for all flip operations flip$(i,j) \in \mathcal{F}_\perp$.*

Corollary 3. *A normal form $\mathcal{F}_\perp$ is sufficiently characterized by an ordered enumeration of all flip operations indices. A normal form $\mathcal{F}_\perp = \{$flip(i,j), flip$(k,l)\}$ may therefore be denoted as $\mathcal{F}_\perp = \{i,j,k,l\}$.*

Every set of flip operations $\mathcal{F}$ is transformable into its normal form $\mathcal{F}_\perp$ by elimination of double flips and flip overlaps and sorting of flip indices.

3.4 Solving occlusions by dynamic programming

This section introduces an algorithm that solves the optimization problem modelled in the previous section using a dynamic programming approach that results in a generalization of the Viterbi algorithm.

The proposed optimization algorithm computes the most plausible identity assignment throughout the entire video by maximizing $\sum s + \sum t$ under the flip operation. Intuitively, this enables s-methods and t-methods to complement and correct each other, especially in cases where an s-method indicates certainty but a t-method does not or vice versa.

The algorithm exploits mathematical properties of the flip operations. When searching for optimal solutions it is sufficient to traverse normal forms of flip operations only. This reduces an infinite search space to an exponential search space. By sorting (commutative, non-overlapping) flip operations in ascending order intermediate results for all flip operations up to a sequence τ_k may be reused. The dynamic programming approach therefore traverses the exponential search space within *linear time* and still guarantees to derive the shortest set of flip operations that is required to transform an arbitrary identifier initialization into the assignment with the highest global plausibility. This enables assignment of local identifiers for flies A$_i$ and B$_i$ to global identifiers 1 and 2 and to sort fly attributes according to global fly identifiers.

Algorithms 1, 2, 3 and 4 below provide a formal definition of the optimization approach.

Algorithm 1 initialize
```
Require: $s, t$
Ensure: $S, T$
  $n \leftarrow |s|$
  $T_{1,-1} \leftarrow -\infty$
  $T_{1,+1} \leftarrow 0$
  for $i = 1$ to $n$ do
    for $c \in \{-1, +1\}$ do
      $S_{i,c} \leftarrow T_{i,c} + c \cdot s_i$ {add score if sign positive, subtract if negative}
    end for
    for $c \in \{-1, +1\}$ do
      $T_{i+1,c} \leftarrow max(S_{i,c} + t_{i+1}, S_{i,-c} - t_{i+1})$ {add to same score or subtract from flipped score}
    end for
  end for
  return $S, T$
```

Algorithm 2 backtrack
```
Require: $S, T, s, t$
Ensure: $flippos$
  $c \leftarrow 1$
  for $i = n + 1$ down to 2 do
    if $T_{i,c} = S_{i-1,c} + t_i$ then
      $flippos_i \leftarrow 0$
    else
      $flippos_i \leftarrow 1$
      $c \leftarrow -c$
    end if
  end for
  return $flippos$
```

Algorithm 3 bulkflip
```
Require: $flippos, s, t$
Ensure: $s', t'$
  $fp_i \leftarrow -(flippos_i \cdot 2 - 1)$ {$-1$ if flippos true, $+1$ otherwise}
  $t'_i \leftarrow t_i \cdot fp_i$ {$-t$ at flippositions, $t$ otherwise}
  $k_i \leftarrow \prod_{j=1}^{i} fp_j$ {$-1$ between flippositions, $+1$ otherwise}
  $s'_i \leftarrow s_i \cdot k_i$ {$-s$ between flippositions, $s$ otherwise}
  return $s', t'$
```

```
**Algorithm 4** optimizeAssignment
Require: $s, t$
Ensure: $s', t'$
  $S, T \leftarrow initialize(s, t)$
  $flippos \leftarrow backtrack(S, T, s, t)$
  $s', t' \leftarrow bulkflip(flippos, s, t)$
  return $s', t'$
```

The final algorithm listed as Algorithm 4 consists of the following steps:

1. A dynamic programming initialization step (see Algorithm 1), where s values and t values are traversed once to compute the cumulative scores S and T, such that $S_{i,c}$ and $T_{i,c}$ contain the best possible scores up to sequence σ_i resp. τ_{i-1} and the condition $c = -1$ that the current sequence is flipped and identifiers are swapped, respectively, $c = 1$ that they remain unchanged. This step exploits the mathematical properties of the flip operation in order to model the optimization problem as a dynamic programming problem instance. The cumulative score up to the first occluded sequence is initialized with $T_{1,-1} = -\infty$ and $T_{1,+1} = 0$. This enforces fly 1 of the global assignment to fulfill the property of positive s values. The total cost of the global identity assignment is given in $T_{n+1,1}$.
2. A backtracking step (see Algorithm 2), where the chosen path that lead to the assignment with best score in $T_{n+1,1}$ is reconstructed. This path determines the flip positions that sufficiently characterize $\mathcal{F}_\perp$, the desired smallest set of flip operations that transforms an arbitrary initialization into the optimal solution.
3. A bulk-flip step (see Algorithm 3), where the result flip operations in $\mathcal{F}_\perp$ are applied to the initially given s and t values in order to derive the flipped scores s' and t' of the optimal solution, $\sum s' + \sum t' = T_{n+1,1}$.

The algorithm result is applied by swapping fly objects within the time series data. For each sequence σ_i, a value $\text{swap}_i = \frac{s'_i}{s_i}$ with $\text{swap}_i \in \{-1, +1\}$ may be computed, in case $\text{swap}_i = -1$ the identifiers for sequence σ_i have to be swapped.

Finally, three minor improvements are suggested: (1) All s and t values v that are 0 are replaced by $v = \epsilon$ where ϵ is the smallest representable floating point number that can carry a sign. This replacement does not affect the algorithm result but instead keeps track of all signs for s and t values and guarantees that all divisions are defined. (2) The maximum impact of a single s or t value should be limited, the current implementation guarantees for machine-generated s or t values v that $\epsilon \leq |v| \leq \Omega$ with $\Omega = 20$. (3) The bulk-flip step may optionally be simplified to compute and return only k instead of s' and t', since k_i is equivalent to swap_i.

The algorithm runs in linear time $\mathcal{O}(m)$ with regards to the total number of sequences $m = |\Phi_0| = 2|\{\tau_i\}| - 1$ and is fast enough for being computed in real time. Manually overruled τ_i sequences are assigned with a t value of $T\_MAX = \Omega \cdot m + 1$ such that machine decisions cannot vote them down and the most plausible global assignment is adapted accordingly.

For occluded scenes, a revised certainty value c_i that resembles the global confidence of the algorithm may optionally be computed. This revised value consists of the known local certainty t_i and a global certainty value ΔT that is computed as a difference between global assignment costs. The algorithm computes the cost to derive an assignment $T'_{n+1,1}$, where t'_i is guaranteed to be set in opposite direction $t'_i = -T\_MAX \cdot \text{sign}(t_i)$ and computes $\Delta T = (T_{n+1,1} - t_i) - (T'_{n+1,1} - t'_i)$. The total confidence c_i of the combined certainty values t_i and ΔT can be expressed as a probability measure, the optional computation of all confidence values runs in quadratic time $\mathcal{O}(m^2)$.

3.5 Experimental results

During our project, we processed more than ten thousand multi-chamber videos containing more than a billion single-chamber frames. Our occlusion methods were tested on 8 randomly selected *Drosophila* courtship videos, each containing 11 chambers with male-female pairs of the same genotype. Each chamber had a diameter of 1 cm and was covered by an anti-reflecting glass plate on top. Videos were recorded from the top at 25 frames per second.

The chamber videos were preprocessed, tracked, postprocessed and manually annotated to establish a ground truth. From our 88 original chambers, 5 were rejected by the preprocessor (wrong number of flies) or due to lack of manual annotation. The remaining 83 chambers contained 8,421 occlusions and 610,919 frames of two-fly behaviour before copulation.

The identity assignment during σ sequences using the Hungarian algorithm turned out to be extremely reliable. We identified potential problems when flies jump (rapidly move to a random new destination, within one frame) and therefore specifically detect such jump events and treat them like occlusions. In particular, identities in sequences before and after the jump event are independently assigned using the Hungarian algorithm and global identities are then assigned using our global occlusion resolvement methods. However, in case two flies jumped exactly to each others place within a single frame this would trick the jump detector and result in an assignment error within the σ sequence. We recorded videos with

25 frames per second and noticed only two such errors during the entire project, which involved tracking about one billion frames. We did not further quantify this error rate due to its rarity and want to denote that recording at higher frame rates would further decrease the error potential.

Figure 5 contains four examples that demonstrate error patterns and how different methods complement each other. The table rows contain alternating σ and τ sequences. The first column contains a sequence identifier; the second column, the length of each sequence in frames. The following three columns contain s values, respectively, t values of s- and t-methods siz1m (deciding based on size differences), posm (deciding based on fly positions) and bocT (deciding based on identifier-containing point sets that are 'carried through' an occlusion).

The remaining six columns contain identifier assignment results produced by different methods. The first three columns contain decisions of local methods only: They are combined with nothing but 'zeros' and therefore analyzed individually for their assignment decisions. For the last three columns, methods were combined with each other. Each assignment entry contains a $< value >$ and a $< decision >$ (separated by a semicolon), the $< value >$ resembles the s or t value associated with the given $< decision >$ identifier assignment. Entries with correct $< decision >$s are colored in green, incorrect assignments are bold and in red. This implicitly encodes the ground truth.

The first example depicts the typical error pattern of local t-methods. The occlusion in $\tau_{41.56}$ is wrongly resolved by methods posm and bocT. Method posm therefore mis-assigns identities for its following σ sequence $\sigma_{41.57}$ *and all* σ *sequences thereafter*, up to the end of the video or another mis-assignment. Apparently, the bocT method already had a mis-assignment before $\tau_{41.56}$ since identifiers were swapped in $\sigma_{41.55}$ and $\sigma_{41.56}$ before occlusion $\tau_{41.56}$. Due to the second mis-assignment, the identifiers are swapped back and result in correct σ sequences after the second mis-assignment.

Having the last three columns in green shows that all three of our combining methods are capable of rescuing this case. The main reasons for the combined method's success are their fundamentally different error patterns. Since combined methods involve both s- and t-methods, a t-method failure may still result in swapped identifiers, but they are typically swapped back immediately since it is not plausible to swap too many σ sequences despite continuous negative evidence coming from the s-method. Examples three and four depict such 'double errors' that are typical for combined methods.

We further want to discuss the robustness of combined methods by examining column *siz1m* : *bocT* in the first example. Although method siz1m assigns the score of ϵ (don't know) in $\sigma_{41.57}$ right after the sequence that bocT

		s-value	t-value	t-value	*siz1m*	*zeros*	*zeros*	*siz1m*	*siz1m*	*siz1m*
seq_{id}	$\lvert seq\rvert$	*siz1m*	*posm*	*bocT*	*zeros*	*posm*	*bocT*	*posm*	*bocT*	*posm, bocT*
$\sigma_{41.55}$	155	$-\Omega$			$\Omega:1$	$-\epsilon:1$	$-\epsilon:0$	$\Omega:1$	$\Omega:1$	$\Omega:1$
$\tau_{41.55}$	1		4.06	ϵ	$\epsilon:0$	$4.06:0$	$\epsilon:0$	$4.06:0$	$\epsilon:0$	$4.06:0$
$\sigma_{41.56}$	14	-9.70			$9.70:1$	$-\epsilon:1$	$-\epsilon:0$	$9.70:1$	$9.70:1$	$9.70:1$
$\tau_{41.56}$	13		1.14	0.77	$-\epsilon:1$	$1.14:0$	$0.77:0$	$-1.14:1$	$-0.77:1$	$-1.91:0$
$\sigma_{41.57}$	1	ϵ			$\epsilon:0$	$-\epsilon:1$	$-\epsilon:0$	$\epsilon:0$	$\epsilon:0$	$\epsilon:0$
$\tau_{41.57}$	1		3.90	2.63	$\epsilon:0$	$3.90:0$	$2.63:0$	$3.90:0$	$2.63:0$	$6.53:0$
$\sigma_{41.58}$	7	4.84			$4.84:0$	$-\epsilon:1$	$-\epsilon:0$	$4.84:0$	$4.84:0$	$4.84:0$
$\tau_{41.58}$	1		4.80	2.63	$\epsilon:0$	$4.80:0$	$2.63:0$	$4.80:0$	$2.63:0$	$7.43:0$
$\sigma_{41.59}$	2	1.10			$1.10:0$	$-\epsilon:1$	$-\epsilon:0$	$1.10:0$	$1.10:0$	$1.10:0$
$\sigma_{45.148}$	68	$-\Omega$			$\Omega:1$	$\epsilon:1$	$-\epsilon:1$	$\Omega:1$	$\Omega:1$	$\Omega:1$
$\tau_{45.148}$	1		2.96	2.55	$-\epsilon:1$	$2.96:0$	$2.55:0$	$2.96:0$	$2.59:0$	$5.51:0$
$\sigma_{45.149}$	10	1.57			$1.57:0$	$\epsilon:1$	$-\epsilon:1$	$-1.57:1$	$-1.57:1$	$-1.57:1$
$\tau_{45.149}$	1		4.09	2.63	$-\epsilon:1$	$4.09:0$	$2.63:0$	$4.09:0$	$2.63:0$	$6.71:0$
$\sigma_{45.150}$	29	$-\Omega$			$\Omega:1$	$\epsilon:1$	$-\epsilon:1$	$\Omega:1$	$\Omega:1$	$\Omega:1$
$\sigma_{45.48}$	123	Ω			$\Omega:0$	$-\epsilon:0$	$-\epsilon:0$	$\Omega:0$	$\Omega:0$	$\Omega:0$
$\tau_{45.48}$	5		2.66	2.63	$-\epsilon:1$	$2.66:0$	$2.63:0$	$-2.66:1$	$-2.63:1$	$5.29:0$
$\sigma_{45.49}$	10	-6.93			$6.93:1$	$-\epsilon:0$	$-\epsilon:0$	$6.93:1$	$6.93:1$	$-6.93:0$
$\tau_{45.49}$	4		2.76	2.46	$-\epsilon:1$	$2.76:0$	$2.46:0$	$-2.76:1$	$-2.46:1$	$5.23:0$
$\sigma_{45.50}$	229	Ω			$\Omega:0$	$-\epsilon:0$	$-\epsilon:0$	$\Omega:0$	$\Omega:0$	$\Omega:0$
$\sigma_{41.47}$	188	$-\Omega$			$\Omega:1$	$\epsilon:0$	$\epsilon:1$	$\Omega:1$	$\Omega:1$	$\Omega:1$
$\tau_{41.47}$	1		-0.59	-0.73	$-\epsilon:1$	$0.59:1$	$0.73:1$	$0.59:1$	$0.73:1$	$-1.32:0$
$\sigma_{41.48}$	3	1.95			$1.95:0$	$-\epsilon:1$	$-\epsilon:0$	$1.95:0$	$1.95:0$	$-1.95:1$
$\tau_{41.48}$	11		1.44	1.93	$-\epsilon:1$	$1.44:0$	$1.93:0$	$-1.44:1$	$-1.93:1$	$3.37:0$
$\sigma_{41.49}$	71	$-\Omega$			$\Omega:1$	$-\epsilon:1$	$-\epsilon:0$	$\Omega:1$	$\Omega:1$	$\Omega:1$

Figure 5 Selected examples and error patterns. Four examples, each depicting alternating σ and τ sequences, σ sequences are on light-gray background. Columns contain a sequence identifier, sequence length (frames), *s* values of *siz1m*, *t* values of *posm*, *t* values of *bocT*. S-methods provide *s* values for σ sequences, t-methods provide *t* values for τ sequences. The remaining six columns contain color-coded $<$ *value* $>$:$<$ *decision* $>$ pairs, the two headers specify which s-method (above) and t-method (below) were combined for each column, combinations with *zeros* resemble local method results. $<$ *decision* $>$ entries indicate resulting identifier assignments for local respective combined methods in each column; green color resembles correct assignments; bold and red color marks wrong assignments. $<$ *value* $>$ entries resemble the local method's certainties; ϵ resembles the smallest possible value and is used when a method cannot come up with a meaningful decision; Ω resembles the largest possible value.

would get wrong and although bocT assigns ϵ in $\tau_{41.55}$, the occlusion right before the troubled occlusion, the combined method still gets the whole assignment right. How is that possible?

In order to mis-assign $\tau_{41.56}$ according to the evidence coming from bocT, the combined method $siz1m$: $bocT$ would have to do a *double*-error. The two most obvious options for that would be to either perform a flip(45.55, 45.56) or a flip(45.56, 45.57) operation. However, the costs for flip(45.55, 45.56) are less attractive than for the no-flip case, $(-(\epsilon) - 9.70 + 0.77) < (\epsilon + 9.70 - 0.77)$. Obviously, the high confidence of siz1m in $\sigma_{41.56}$ makes this option unattractive, and similarly for flip(45.56, 45.57), $(0.77 - \epsilon - 2.63) < (-0.77 + \epsilon + 2.63)$. In this case the higher score for $\tau_{41.57}$ coming from method bocT itself makes the difference. Flipping even longer sequences, e.g. flip(45.56, 45.58) would be even less attractive for the algorithm. The most plausible identifier assignment is determined correctly - despite wrong evidence coming from bocT in $\tau_{41.56}$ and two proximate ϵ values in $\tau_{41.55}$ and $\sigma_{41.57}$.

The second example depicts a similar case, this time method siz1m mis-assigns $\sigma_{45.149}$, but methods posm and bocT both get this case right. Again, all combined methods come up with the correct assignment as the combined evidence coming from posm or bocT is stronger than the misleading evidence from siz1m.

In the third example, in sequence $\sigma_{45.49}$ method siz1m is wrong and rather confident about it. In this case both combined methods $siz1m$: $posm$ and $siz1m$: $bocT$ would fail too, however, method $siz1m$: $posm, bocT$ which uses the stronger Dempster-combined evidences from posm *and* bocT is still capable of coming up with the correct assignment.

The last example shows a case where $siz1m$: $pos, bocT$ is wrong. Although methods siz1m, $siz1m$: $posm$ and $siz1m$: $bocT$ would solve the case correctly, the combined wrong evidences of posm and bocT outweigh the value coming from siz1m. The video frames of this error instance are depicted in Figure 3d.

Table 1 summarizes our performance evaluation where all identifier assignment methods introduced in Section 3 were applied to our annotated test set. The table compares local methods and different combined methods, named according to the scheme $< s - method >:< t - method >$. Again, a combination with zeros is used to quantify local methods only. All evaluated methods are compared according to two quality measures: (a) the percentage of correct assignments of identifiers before an occlusion to identifiers after that occlusion and (b) percentage of correctly assigned frames in unoccluded sequences.

Methods in Table 1 rows 1 to 4 involve t-methods only. Aside from the local methods posm and bocT, we further evaluated meta-method CART, a machine learning method that uses a classification tree to come up with an assignment based on multiple t values and occlusion properties like an occlusions length or its maximum overlap. Note that CART is still a t-method as it combines multiple t-methods. We provide results for overfitted CART_O and cross-validated CART_C, where 10-fold cross-validation was applied.

Table 1 Performance summary

	S-method : t-method	Correct τ (%)	Correct σ-frames (%)
1	zeros : posm	94.28	51.73
2	zeros : bocT	91.73	53.90
3	zeros : CART_O	99.75	87.32
4	zeros : CART_C	98.96	73.94
5	siz1m : zeros	93.37	99.74
6	siz1m : posm	98.86	99.88
7	siz1m : bocT	99.17	99.95
8	siz1m : posm, bocT	99.55	99.97
9	siz1m : posm $\sim$	99.47	99.95
10	siz1m : posm $\sim$, bocT $\sim$	99.62	99.97
11	siz1m : CART_O	99.9169	99.99
12	siz1m : CART_C	99.1331	99.92

Left column: <s-method>:<t-method>; middle column: occlusion accuracy as correct τ percentage; right column: frame accuracy as correct σ-frames percentage.

Although the accuracy of local methods for correct occlusions are 94.28% and 91.73%, the methods get only 51.73% and 53.90%, respectively, of unoccluded frames correct. This is due to the error propagation problem that is outlined in the first example of Figure 5. As expected, the CART approach alone cannot overcome this problem. Although combined t scores lead to a highly improved occlusion accuracy, the t-intrinsic error pattern still leads to low frame accuracy.

In row 5 the size-based s-method is evaluated. Although it comes with similar occlusion accuracy as the t-methods, its frame accuracy is highly improved. This is because incorporation of s-methods leads to double-error patterns where wrong occlusion assignments are immediately swapped back. Therefore, such methods typically get only single σ sequences wrong.

All further rows 6 to 12 contain performance values for combined methods. In 6 to 8 methods $siz1m$: $posm, siz1m$: $bocT$, and $siz1m$: $posm, bocT$ show the impact of the dynamic programming approach to the combined methods performance. As shown in the examples in Figure 5 above, s-methods and t-methods complement and correct each other. Despite their double-error patterns that minimize the number of mis-assigned σ

sequences, combined methods further minimize the length for mis-assigned σ sequences. The *s* value coming from siz1m is designed to be dependent on the lengths of observable σ sequences, such that long sequences (on which the method performs well) are given high scores and utterly short sequences (where the method sometimes is wrong) are given low scores. Typical error pattern for combined methods are therefore double errors that contain single *short* sequences, typically consisting of one or two frames, which explains the high frame accuracy of these combined methods.

We further evaluated the combination of methods of same type using the Dempster combination [17,18] and it turned out that the use of Dempster-combined t-method $posm, bocT$ in row 8 slightly outperformed simpler dynamic programming combinations in rows 6 and 7.

Finally, the methods evaluated in 9 to 12 turned out to result in little or no improvements. In Section 3.3 we mention that probability values are derived from method scores using a linear approximation. In 9 and 10, we evaluate combinations with methods posm$\sim$ and bocT$\sim$ where nonlinear approximations are used to derive more precise probability values. However, it turned out that these performance improvements between $siz1m : posm, bocT$ and $siz1m : posm \sim, bocT \sim$ corrected only nine more frames.

When combining s-methods with CART-methods, it turns out that the overfitted method $siz1m$: CART$_O$ outperforms all other methods; however, the cross-validated method $siz1m$: CART$_C$ shows a decrease in performance. This is mostly because the CART-method typically returns very confident scores that are difficult to be corrected by other methods. The method CART$_C$ is a good example for a t-method that outperforms other t-methods in occlusion accuracy (*cf.* Table 1: rows 1, 2 and 4), but still is outperformed in terms of frame accuracy due to a lack of combinability (*cf.* Table 1: 8 and 12).

4 Other application: resolving heading

A fly body or an ellipse covering a fly body consists of two *ends* A and B where the flies' axis crosses the flies' perimeter. Resolving the heading problem means to find out whether end A or end B is the flies head.

Fortunately, there are several evidences from the flies' anatomy and behaviour. First, flies typically walk in a forward direction. The movement direction of a fly may be used to predict at which side to find the head. Secondly, the flies' wings typically point in backwards direction. Therefore, vector from Centroid to wCentroid may be used as a second independent predictor. Finally, the head does typically not flip by 180° within a single frame.

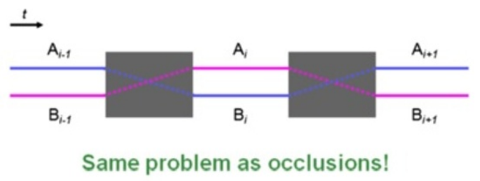
Interestingly, the heading problem may be reduced to the occlusion problem described in Section 3 and the proposed optimization algorithm of Section 3.3 may be applied to solve the heading problem as well.

The idea is to model every single frame as a σ sequence and 'artificial gaps' between frames as τ sequences. The evidences from movement and wings are incorporated as s-methods (again s-methods have to operate on attributes that are comparable through the entire video) and a known persistence constraint is incorporated as a t-method. The computed *s* values correspond to probabilities for point A being the head of the fly, and *t* values correspond to probabilities that point A in the frame before τ is again point A in the frame after τ.

Figure 6 depicts how to model the heading problem as a problem instance of the same optimization problem that previously solved the occlusion problem. Note the strong similarities between Figures 4 and 6; the text written in red in Figure 6 marks the few differences.

In order to resolve the heading it is sufficient to define the s- and t-methods that incorporate movement, wing anatomy and persistence evidences, and then re-use the very same algorithm and framework as for occlusions.

For this reason, the coordinates of the two endpoints *A* and *B* (after heading assignment called *Head* and *Tail*) and



- For each frame i,
  calculate the probability s_i that A is the head [based on movement and wings],
- Calculate the probability t_i that A_i is A_{i-1}
- Compute most plausible heading assignment by maximizing $\Sigma s + \Sigma t$

Figure 6 Modeling heading resolvement as instance of the same optimization problem. Text in red marks changes to occlusion problem instance.

the Centroids of the body and the wing regions C and W are determined for every frame.

Definition 8. Let X_i denote the value of point X in frame i and $\overline{XY}$ denote the euclidean distance between points X and Y. The s score *score*$_{move}$ is defined as $\text{score}_{\text{move}} = \frac{\overline{A_iC_{i-1}}-\overline{B_iC_{i-1}}}{\overline{A_iC_{i-1}}+\overline{B_iC_{i-1}}}$.

Definition 9. The s score *score*$_{wing}$ is defined as $\text{score}_{\text{wing}} = \frac{\overline{AW}-\overline{BW}}{\overline{AW}+\overline{BW}}$.

Definition 10. The *combined s score* is defined as $\text{score}_{\text{move}\otimes\text{wing}} = \max(\text{score}_{\text{move}}, \text{score}_{\text{wing}})$.

The score $\text{score}_{\text{move}}$ will be positive whenever the fly moved rather in A- than in B-direction, the score $\text{score}_{\text{wing}}$ will be positive in case A is closer to the centroid of the wing region. Note that $-1 \leq \text{score}_{\text{move}} \leq +1$ and $-1 \leq \text{score}_{\text{wing}} \leq +1$. Both scores are combined by a simple maximum aggregation. From the combined score $\text{score}_{\text{move}\otimes\text{wing}}$ probability approximations and finally logodd values s may be derived as in the occlusion case.

Definition 11. The t score *score*$_{persist}$ is defined as $\text{score}_{\text{persist}} = \frac{\overline{AB}+\overline{BA}-\overline{AA}-\overline{BB}}{\overline{AB}+\overline{BA}+\overline{AA}+\overline{BB}}\text{Eccentricity}$.

The persistence score $\text{score}_{\text{persist}}$ is $-\text{Eccentricity} \leq \text{score}_{\text{move}} \leq +\text{Eccentricity}$, with $0 \leq \text{Eccentricity} \leq 1$. Rescaling the score by the Eccentricity attribute ensures low persistence scores when flies 'turn upwards' and are thus round. From such a position, flies may abruptly change their heading via z-direction.

The optimization algorithm (see Algorithm 4) will compute the most plausible *heading* assignment for all video frames by maximizing $\sum s + \sum t$ under the flip operation introduced in the Section 3.3.

For the heading case, the linear time property of algorithm is essential since heading is typically computed for $15,000+$ frames and other, e.g. quadratic algorithms would already become unhandy for these problem instances.

A performance evaluation on 42,870 manually annotated heading situations resulted in 99.2% of correct heading assignments. This number fits to the 'occlusion accuracy' quality measures that we observed for occlusion problem instances in Table 1 (middle column). The other quality measure in that table is not applicable for heading problem instances.

Typical heading error instances are sequences where flies actually *do* walk backwards for a longer time, e.g. due to a series of evasive maneuvers.

5 Conclusions

5.1 Summary and discussion

This paper introduces a two-fly tracker that provides solutions for all major challenges in insect tracking: arena detection, segmentation, resolving occlusions, resolving heading and event detections. It was designed to handle legacy videos that were originally not recorded for automated quantification and therefore provides several methods for quality boosting and quality control (see Section 2). The system automates all attention-critical parts. It checks video quality standards, detects arenas and automatically computes video background, and computes thresholds for body and wing segmentation and a global identity assignment across occlusions. It is therefore designed to minimize required user interactions which makes it suitable to support analysis of large-scale experiments. Compared to manual behaviour scoring, it provides high-throughput, robust and objective quantification for courtship behaviour videos; overcomes lacks of behaviour quantification; and enables systematic identification of genes and neurons that are critical for this behaviour. This may lead to better understanding of how nervous systems regulate stereotypic behaviours. Compared to existing tracking softwares, our system is capable to deal with quality issues that result from video contamination or human error and automatically boosts video quality beyond possible recording conditions. We put huge effort into the occlusion problem and automatically identifying flies without any requirements of marking them. Our system is not limited to specific behaviours and computes a large number of attributes for both identified flies that encourage definition or training of behaviour classifiers. It comes with machine-learned and user-defined classifiers for *Drosophila* courtship behaviour and its sub-behaviours (*cf.* Figure 7). The software includes tools for result inspection and bulk submission of videos to a computer cluster.

The identification of the two flies through the entire video was essential for the detection and assignment of biologically relevant events and the resolvement occlusions with highest possible accuracy turned out to be of particular importance. Since manual correction of occlusion assignments requires lots of user interaction (and user attention), some efforts were invested to come up with an automated solution for that problem (see Section 3).

Section 3.2 introduced different approaches for solving single occlusions. These methods are called t-methods or *local* methods as they focus on individual occlusions without further consideration of their context. Each local method suggests an identifier assignment for each occlusion case and gives a certainty score for its decision. Decisions and scores of multiple local methods may be combined in a meta-method; machine learning based

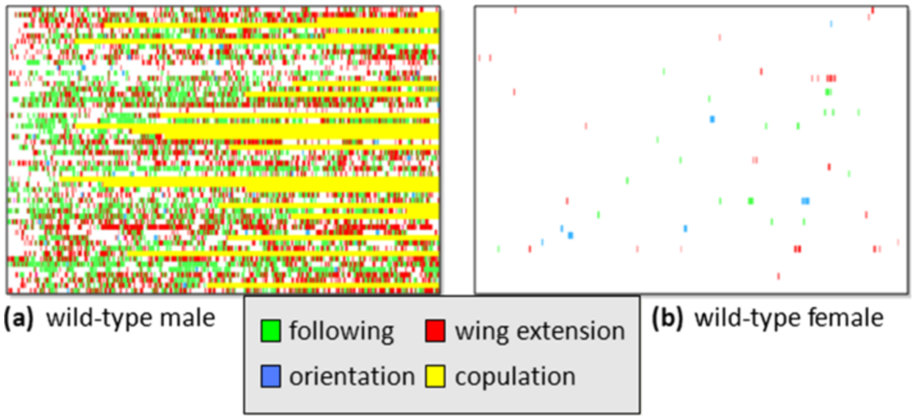


Figure 7 Automatically generated ethograms. Behaviour visualization for 54 flies in 6 min by color coding of identified courtship sub-behaviour. **(a)** Wild-type male behaviour. **(b)** Wild-type female behaviour.

meta-methods may further include observable features further characterizing occlusions.

An intrinsic problem of t-methods or combinations of them comes with their limited local perspective that causes mis-assignments to be propagated until the end of the video. Section 3.3 introduces an optimization approach that incorporates context information. It complements t-methods by s-methods that base on characteristics of unoccluded sequences that are comparable during the whole video. The certainty scores of s- and t-methods are turned into logodd values that are comparable to each other such that the most plausible identity assignment for the entire video is achieved by maximizing the sum of these logodd values under a flip operation (*cf.* Section 3.3). The optimization algorithm introduced in Section 3.4 solves that optimization problem in linear time.

Within a test set of 11 male-female courtship videos with manually annotated ground truth, the introduced local methods scored 90% to 95% of the cases correctly, the combining meta-methods correctly assigned about 99% and the optimization approach assigned up to 99.62% of occlusions correctly. When it comes to correctly assigned identities in unoccluded frames, naive meta-methods suffer from the error propagation problem while the optimization approach improves its accuracy to 99.97%. In other words, from about 6 h and 45 min of the unoccluded video frames, the frames with wrong identity assignment are together about 7 s.

These results are achieved as the algorithm implicitly minimizes the number of mis-assigned unoccluded frames. This property is inherited from the approximation of the sign test that is used as s-method.

Further, the algorithm 'self-corrects' its mistakes. Potential errors typically occur *pair-wise*, one real error immediately followed by a second one that compensates the first error, as it is *not* plausible that identities are wrong from a given point to the end of the video. Wrongly assigned identities are therefore a *local* problem and do not affect the rest of the video.

These two error patterns, pair-wise errors ensuring *local* mis-assignments only and *short* non-occluded sequences as potential error domains, explain the low number of unoccluded frames that are mis-identified and result in desirable properties of the algorithm.

In many occasions, users accidentally *downgraded* the quality of ground truth that was previously automatically pre-annotated. This suggests that the automated occlusion resolvement method may, in many cases, be more reliable than a human annotator. However, the automatically derived occlusion assignments may still be manually inspected and overruled, and an annotating user may sort occlusions by machine-given confidence values.

Another desirable property of the algorithm is that it runs very efficiently (in linear time $\mathcal{O}(|\Phi_0|)$). This enables its applications on large problem instances. Section 4 describes how heading assignments can be modelled as an instance of the very same optimization problem. The very same optimization algorithm then derives the most plausible heading assignment for every frame.

5.2 Future work

We have a system that identifies flies and extracts various attributes; we implemented more than 1,000 automatically observable shape and constellation descriptors. The system currently comes with classifiers for courtship behaviour and its sub-behaviours that allow to visualize observed behaviours as automatically generated ethograms.

Figure 7 depicts an ethogram that visualizes courtship events for wild-type males and females. Our current courtship classifiers are sex-specific (compare left vs.

right) and deliver expected results for known mutants (*cf.* [10]).

We aim to support definition and training for classifiers that capture further meaningful behaviours.

The overall system is currently being transcoded from Matlab to C++ and is optimized for performance such that it runs on a standard laptop within a reasonable computation time.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]I.M.P. Research Institute of Molecular Pathology, Dr. Bohr-Gasse 7, 1030 Vienna, Austria. [2]Present address: University of Oxford, Department for Computer Science, Wolfson Building, Parks Road, Oxford OX1 3QD, UK. [3]University of Vienna, Fachdidaktik- und Lernforschungszentrum Informatik, Währinger Strasse 29, 1090 Vienna, Austria.

Received: 19 February 2013 Accepted: 31 October 2013
Published: 17 December 2013

References
1. BS Baker, BJ Taylor, JC Hall, Are complex behaviors specified by dedicated regulatory genes? Reasoning from *Drosophila*. Cell **105**, 13–24 (2001)
2. BJ Dickson, Wired for sex: the neurobiology of *Drosophila*, mating decisions. Science **322**, 904–909 (2008)
3. JT Pierce-Shimomura, M Dores, SR Lockery, Analysis of the effects of turning bias on chemotaxis in *C. elegans*. J Exp Biol. **208**(Pt 24), 4727–4733 (2005)
4. CJ Cronin, JE Mendel, S Mukhtar, YM Kim, RC Stirbl, J Bruck, PW Sternberg, An automated system for measuring parameters of nematode sinusoidal movement. BMC Genet. **6**(1), 5 (2005)
5. Z Feng, CJ Cronin, JH Wittig Jr, PW Sternberg, WR Schafer, An imaging system for standardized quantitative analysis of *C. elegans* behavior. BMC Bioinformatics **5**, 115 (2004)
6. JH Baek, P Cosman, Z Feng, J Silver, WR Schafer, Using machine vision to analyze and classify *Caenorhabditis elegans* behavioral phenotypes quantitatively. J Neurosci. Methods. **118**(1), 9–21 (2002)
7. JR Martin, A portrait of locomotor behaviour in *Drosophila* determined by a video-tracking paradigm. Behav Processes. **67**(2), 207–219 (2004)
8. H Dankert, L Wang, ED Hoopfer, DJ Anderson, P Perona, Automated monitoring and analysis of social behavior in *Drosophila*. Nature Methods **6**, 297–303 (2009)
9. K Branson, AA Robie, J Bender, P Perona, MH Dickinson, High-throughput ethomics in large groups of *Drosophila*. Nature Methods **6**, 451–457 (2009)
10. C Schusterreiter, Computational analysis of *Drosophila* courtship behaviour. Thesis, University of Vienna (2011)
11. JC Simon, MH Dickinson, *A new chamber for studying the behavior of Drosophila*. PLoS One **5**(1), e8793 (2010)
12. SC Hoyer, A Eckart, A Herrel, T Zars, SA Fischer, SL Hardie, M Heisenberg, Octopamine in male aggression of *Drosophila*. Curr. Biol. **18**(3), 159–167 (2008)
13. PP Ramdya, T Schaffter, D Floreano, R Benton, Fluorescence Behavioral Imaging (FBI) tracks identity in heterogeneous groups of *Drosophila*. Plos One **7**(11) (2012)
14. HW Kuhn, The Hungarian method for the assignment problem. Naval Res. Logistics Q. **2**(1–2), 83–97 (1955)
15. P Gabriel, J Verly, J Piater, A Genon, The state of the art in multiple object tracking under occlusion in video sequences. Advanced Concepts for Intelligent Vision Systems (2003), pp 166–173
16. F Aurenhammer, Voronoi diagrams—a survey of a fundamental geometric data structure. ACM Comput. Surv. (CSUR) **23**(3), 345–405 (1991)
17. AP Dempster, Upper and lower probabilities induced by a multivalued mapping. Ann. Math. Stat. **38**(2), 325–339 (1967)
18. G Shafer, *A Mathematical Theory of Evidence* (Princeton University Press, 1976)

doi:10.1186/1687-5281-2013-64
Cite this article as: Schusterreiter and Grossmann: **A two-fly tracker that solves occlusions by dynamic programming: computational analysis of *Drosophila* courtship behaviour.** *EURASIP Journal on Image and Video Processing* 2013 **2013**:64.