DISSERTATION

Titel der Dissertation

Computational Analysis of Drosophila Courtship Behaviour

Verfasser

Dipl.-Ing. Mag. Christian Schusterreiter

Angestrebter Grad:

Dr. techn.

Wien, 2011

Studienkennzahl lt. Studienblatt:	A 084 881
Studienrichtung lt. Studienblatt:	Dr.-Studium d. Sozial- u. Wirtschaftswiss. Informatik
Betreuer:	Univ-Prof. Dr. Wilfried Grossmann

Abstract

The fruit fly Drosophila melanogaster is a well established model organism for molecular biology and neuroscience studies. It comes with a large set of genetic tools and well preserved innate behaviours, in particular courtship behaviour is a robust and sexually dimorphic behaviour which enables anatomical and functional studies of neuronal circuits.

This thesis is about an automated quantification for functional behaviour studies, it enables measurement and visualization of differences in behaviour for flies with genetically manipulated neurons and neuronal circuits.

Since geneticists typically do large numbers of experiments the aim was to automate the analysis process for a behaviour screen. There are multiple benefits gained from an automated tool like saving time, limiting human error and extending possibilities of analysis. Automated quantification of observed behaviour also provides robust and therefore reproducible analysis.

The quantification itself comes in two steps, an image processing step where recorded fly videos are translated into a time series and a pattern recognition step which searches for known or learned patterns within that time series.

The proposed system offers solutions for all key challenges that were encountered for automated quantification, in particular for arena detection, video quality control, fly segmentation, occlusion resolvment, heading resolvement and event detection. Especially resolving occlusions turned out to be an important but difficult task, therefore a lot of energy was invested to attack that particular challenge.

The result is a fully automated system containing a supervised learning framework that allows to train and apply bottom-up classifiers, e.g. for courtship behaviour, and sex-specific and fru-dependent top-down classifiers that automatically identify individual courtship steps.

The system is designed to minimize user interaction and therefore performs all involved video processing and analysis steps in a fully automated way, it therefore enables a robust, objective and high throughput analysis of large amounts of video data.

The modular structure of the system allows generating spin-off trackers that reuse the system's image processing part for different downstream analysis parts that may be specifically designed or adapted for new biological assays. For the standard courtship assay the system automatically computes ethograms in order to visualize and quantify observed courtship behaviour.

Abstract

Die Taufliege Drosophila melanogaster ist ein weitverbreiteter Modellorganismus für Studien in Molekularbiologie und Gehirnforschung. Man assoziiert sie mit einer grossen Ansammlung von Werkzeugen fuer genetische Manipulationen und mit stabilem angeborenem Verhalten, insbesonders das Balzverhalten ist ein robustes und geschlechtsspezifisches Verhalten welches anatomische und funktionale Analysen von neuronalen Schaltkreisen ermöglicht.

Die Arbeit in dieser Dissertation stellt ein eine automatische Quantifizierung für funktionale Verhaltensanalysen vor die es ermöglicht Verhaltensunterschiede zwischen Fliegen mit genetisch manipulierten Neuronen oder neuronalen Schaltkreisen zu messen und zu visualisieren.

Da Genetiker typischerweise eine grosse Anzahl von Experimenten durchführen wurde die Automatisierung des Analyseprozesses von Serien von Verhaltensexperimenten angestrebt. Ein automatisches Analysewerkzeug bringt viele Vorteile, es spart Zeit, reduziert menschliche Fehler und bringt eine Erweiterung der Analysemöglichkeiten. Eine automatische Quantifizierung von beobarchtetem Verhalten bringt weiters robuste und reproduzierbare Analysen.

Die Quantifizierung selbst erfolg in zwei Schritten, einem Bildverarbeitungsschritt in welchem aufgezeichnete Verhaltensvideos in eine Zeitreihe uebersetzt werden und einem Mustererkennungsschritt in welchem eine solche Zeitreiche nach bekannten oder gelernten Mustern durchsucht wird.

Das vorgestellte system bietet Lösungen fuer alle Schluesselherausforderungen einer solchen automatischen Quantifizierung, im speziellen für eine automatische Arenadetektierung, einen automatische Qualitätskontrolle für Videos , die Segmentierung der Fliegen, das Auflösen bzw. Zuordnen von Überdeckungen, das Identifizieren des Kopfendes und das Detektieren von biologisch relevanten Ereignissen.

Speziell das Auflösen von Überdeckungen hat sich als eine wichtige, aber schwierige Aufgabe herausgestellt, es wurde viel Energie investiert um auch diese Herausforderung in Angriff zu nehmen und zu lösen.

Das Ergebnis ist ein vollautomatisches System welches einen Rahmen fuer supervised learning beinhaltet, welcher das Trainieren und Anwenden von bottom-up Klassifikatoren, z. B. für Balzverhalten, ermöglicht. Das System beinhaltet weiters geschlechtsspezifische und *fru*-abhängige top-down Klassifikatoren welche die automatische Identifikation von einzelnen Balz-Schritten ermöglichen.

Das System wurde so entworfen dass Benutzerinteraktionen minimiert werden, es arbeitet alle involvierten Bildverarbeitungs- und Analyseschritte vollautomatisch ab und erlaubt daher eine robuste und objektive Analyse mit hohem Durchsatz, und ist somit anwendbar für die Analyse grosser Mengen von Videodaten.

Das modulare Design des systems erlaubt weiters das Entwickeln von Spin-offs, welche den Bildverarbeitungsteil wiederverwenden und darauf aufbauend Verhaltensanalysen berechnen, die speziell auf andere Biologische Experimentumgebungen oder neue Verhaltensmuster abgestimmt sind. Im Hauptanwendungsfall, in dem das Balzverhalten der Drosophila melanogaster analysiert wird, werden automatisch Ethogramme generiert, die das beobachtete Balzverhalten quantifizieren und visualisieren.

Contents

1 Introduction **6**

1.1 Courtship as Behaviour Model 6
1.2 Previous Methods of Analysis 7
1.3 Setting and Goals 7
1.4 Related Work 10

2 System Overview **11**

3 Preprocessor **15**
3.1 Workflow 16
3.2 Methods 21
3.2.1 Background Extraction 21
3.2.2 Illumination and Color Correction 26
3.2.3 Arena Detection 31
3.2.4 Quality Control Checks 38
3.2.4.1 Checking Flies per Chamber 38
3.2.4.2 Movement Detection and Boundary Intrusions . 39
3.2.4.3 Other 43

4 Tracker **47**
4.1 Workflow 47
4.2 Methods 48
4.2.1 Segmentation 48
4.2.1.1 Thresholding 49
4.2.1.2 Thresholding Discussion 58
4.2.1.3 Thresholding Combined With Gradient Correction 59
4.2.2 Foreground Refinement 65
4.2.3 From Regions to Flies 68
4.2.4 Primary Attribute Extraction 71

5 Postprocessor **73**
5.1 Workflow 73
5.2 Methods 77
5.2.1 Identity assignment 77
5.2.1.1 Occluded and Non-occluded Sequences 77
5.2.1.2 Identity Assignment for σ Sequences 79
5.2.1.3 Identity Assignment for τ Sequences 80
5.2.2 Resolving Occlusions 81
5.2.2.1 Approach I: Combination of t-Methods 82
5.2.2.2 Approach II: Combination of s and t-Methods . 110
5.2.3 Other Methods 122
5.2.3.1 Checking Frame Validity 122
5.2.3.2 Handling Manual Annotations 122

5.2.3.3 Resolving Heading 124
5.2.4 Interpolation Across Occlusions 128
5.2.4.1 Interpolating Ellipse Characteristics 129
5.2.4.2 Wing Extensions During Occlusions 131
5.2.5 Event Detection 135
5.2.5.1 Bottum-up Classification 136
5.2.5.2 Top-down Classification 142

6 Tools **158**
6.1 webInterface 158
6.2 annotationTool 164
6.3 Spin Offs 168
6.3.1 Spin Off 1: Chaining Detector 168
6.3.2 Spin Off 2: Food Preference Analysis 169
6.3.3 Spin Off 3: Analysis of Wing Extension Direction 170

7 Conclusion **174**

Appendix: Preliminary Definition and Notation **180**

List of Figures **182**

References **191**

Acknowledgements **194**

1 Introduction

1.1 Courtship as Behaviour Model

How is innate behaviour encoded in genes? A fundamental question in neuroscience is to understand the relation between genes, brains and behaviour: While genes encode hard-wired neuronal circuits within a nervous system, such neuronal circuits may produce motor output resulting in observable behaviour.

One approach to this question is to use transgenic model organisms in order to study anatomical and functional phenotypes, such genetic studies may identify local gene expressions whose behavioural phenotypes reflect roles in nervous system functions.

The fruit fly Drosophila melanogaster has a set of characteristics that makes it a very attractive object of study in neurobiology, especially in order to study how the nervous system generates behaviour. It initiates behaviours that are neither learned nor require any preliminary experience, which suggests that they are genetically controlled and more or less hard-wired to the nervous system. Several innate behaviours of Drosophila melanogaster are sex-specific. In combination with the availability of genetic and molecular tools, it is an ideal model to study how the nervous system generates behaviour.

Drosophila courtship behaviour is a very robust, sex-specific and innate behaviour, given available genetic tools a robust quantification of this behaviour will help to systematically identify involved genes and neurons. Courtship behaviour further is a well analyzed innate behaviour, the gene that regulates whether Drosophila melanogaster shows the typical male or female courtship behaviour has already been identified.

In particular it was shown that "male courtship requires products of the fruitless (fru) gene, which is spliced differently in males and females" [2], that male splicing is necessary for male courtship behaviour and that "male splicing is also sufficient to generate male behavior in otherwise normal females. These females direct their courtship toward other females (or males engineered to produce female pheromones)." [2] It was a big surprise that a complex behaviour like courtship is regulated by a single neuronal gene. It is strongly believed that this gene interacts with cascades of downstream genes that regulate individual parts of courtship behaviour. Some aspects of courtship have already been shown to be dependent on such downstream genes affecting e.g. specific olfactory neurons [3] or the female post-mating switch [4]. However, many parts of the downstream puzzle characterizing the involved genetic and neuronal circuits may still be missing.

For all these reasons the fruit fly Drosophila melanogaster is a heavily studied model organism and its courtship behaviour is of particular interest in order to explore how behaviour is encoded in genes.

1.2 Previous Methods of Analysis

The main goal of automated vision processing in behaviour studies is to quantify and classify different behaviours. A short survey through description of courtship behaviour within the past 50 years shows that descriptions were further and further simplified. In the 1950s ethologists manually captured complex courtship patterns within ethograms, later in 2001 complexity of courtship was simplified to six sequential steps [1], until it was finally (by geneticists) reduced to a single number, the courtship index, which is the simply the percentage of time a fly courts [5].

Interestingly, scoring this single number was still considered to be a too time-consuming and tedious task. Geneticists do large numbers of experiments, and the work load of manual quantification, especially when it comes to genome-wide behaviour screens, became a limiting factor for them.

1.3 Setting and Goals

The above constraint, geneticists do a large number of experiments and analysis of such behaviour screens requires time consuming manual assessment, motivated the aim to automate this process as much as possible. Furthermore, there are multiple more benefits gained from an automated tool. Saving time is an important factor, but of course automation also limits human error and extends possibilities of analysis. Automation also provides robust and therefore reproducible analysis.

I therefore want to come up with an automated, and thus high-throughput, robust and objective method that captures the whole complexity of courtship behaviour and translates given courtship behaviour videos by computer vision and statistical methods into rich ethogram-like descriptions. The automated quantification will first translate a video into a time series, then search for known patterns in the time series data. In order to capture more complex courtship patterns, I aim to identify individual flies through the entire video and then generate an automated ethograms for each of the flies.

The setting comes with two interdisciplinary interfaces: First, the input interface, which is defined to accept videos and second the output interface, which generates *ethograms*, a protocol and visualization of observed behaviours. The main purpose of the overall system is to transform one into the other, namely videos into ethograms, it thus accepts high amounts of video data containing biological information and transforms it into low amounts of biological relevant data with high information density.

Figure 1 shows a snapshot of a sample input video. A video typically records the behaviour in multiple *chambers* or *arenas* (in figure 1 the regions marked in green), within such arenas the flies are observed.

Within the standard setting multiple round courtship chambers, each containing exactly two flies are analyzed[1]. A courtship chamber typically contains

[1] See sections 6.3.1 and 6.3.2 for a short description of other scenarios.

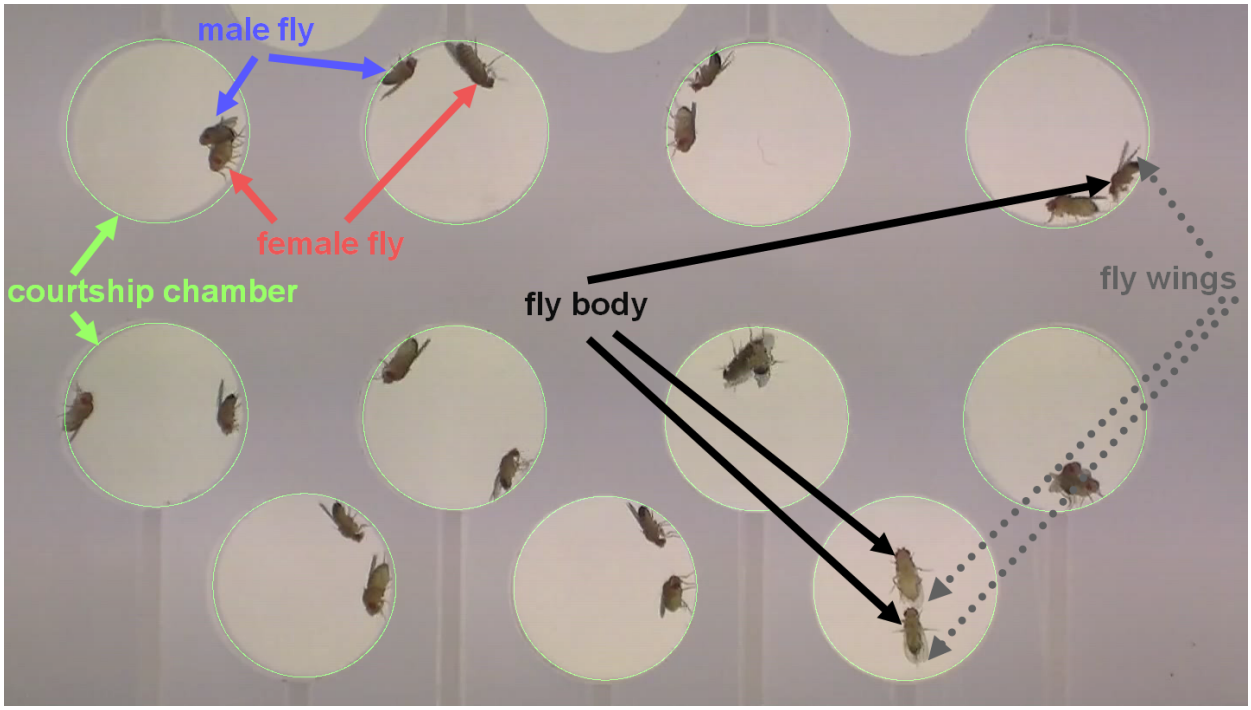


Figure 1: Snapshot of a courtship video. The video contains several courtship chambers (green), each containing a male and a female fly (blue resp. red arrows). Both flies morphologically consist of two regions, a dark body region and a brighter wing region (black resp. gray arrows).

a male and a female fly. While a wildtype[2] female fly is typically slightly bigger (see figure 1, annotation in red and blue) the wildtype male fly takes the active role in the courtship interplay.

The analysis is performed for every single chamber and comes in two steps.

At first videos are transformed into a time series of trajectories that contain detailed information about fly positions and fly *morphologies*, a fly consists of *bodies* and *wings*. The snapshot in figure 1 also contains flies, the dark regions are the fly body regions, the brighter wing regions are typically on the back of the flies (see black and gray annotations in figure 1).

From these trajectories further and advanced characteristics, like the distance between the flies or fly identities, may be derived and transformed into the ethogram descriptions, that capture the desired biologically relevant information.

[2] The term *wildtype* stands for non-mutant, and refers to genetically unmodified laboratory flies, named *Canton-S*.

The above two steps**,** transformation of videos into time series and of time series into ethograms, come with a variety of problems and challenges. A main challenge is that videos are recorded in different settings and with different quality. Although biologists aim to record videos under same conditions, they have to overcome lots of practical issues to be able to do so, e.g. they share equipment for different experiments and therefore may have to reproduce the recording setting from scratch. Further, equipment is evolving and cameras may change. And finally, also a laboratory is not always a perfect world, which may cause unexpected effects or human error.

The image processing step therefore has to deal with variations in the setting which may result from different lighting conditions, sometimes uneven lighting within a single video, with different contrasts and with videos recorded with different camera zooms. Further some recorded arenas might be contaminated and flies from outside may cross the recorded scenes unexpectedly. In order to deal with this quality standards are defined to ensure functionality and the system aims to check whether these standards are met. It is required that chambers are neither moved nor intruded and that they do not overlap each other or with the border of the video, such that flies can't hide outside the recording area. Further, a clearly visible contrast between fly bodies and fly wings is required. However, uneven lighting conditions and different camera zooms can both be automatically detected and compensated by the system.

For the behaviour analysis it is necessary to track *and identify* the individual flies through the entire video. Since the two flies sometimes heavily interact and therefore *occlude* each other, which mean that they overlap each other such that only one fly region is visible[3], identity assignment turned out to be a major challenge. A tool to manually overrule automatic assignments is provided, however, machine assignments are very reliable and manual inspection is typically not required.

Detecting courtship events and discriminating sub-behaviours turned out to be a challenge, especially advanced characteristics with high, biologically relevant information content were difficult to determine. Particularly detecting the positions of the flies heads turned out to be difficult and useful here. When the project was started no formal description characterizing courtship behaviour as a whole or in its steps was available. However, biologists had the ability to judge whether and when courtship behaviour is observed. One aim of the project was to turn this implicit knowledge into an explicit characterization, which allows robust assessment of behaviour and comparison of data from different persons or labs. Further, a formal characterization is a required basis for an automated quantification.

The goal of my project is to develop an automated system and methods that attack the problems and challenges mentioned above and to come up with an automated method that can assess courtship behaviour in videos, and therefore

[3] Figure 1 shows two flies occluding each other in the arena in the top left corner (copulating flies), and in the arena in the third column, second row (random encounter).

allows to automatically measure a courtship index and further captures the complex behaviour in all its details.

1.4 Related Work

When the project was initiated only a few basic worm trackers [24][25][26][27] for different model organism called *Caenorhabditis elegans* existed. These trackers mainly analyzed the worms movements, they quantified turn direction versus straight movement and excluded frames where worms occluded each other. The only published tracker for Drosophila was [28] which was used to analyze Drosophila locomotion behaviour.

During a day visit in the Heisenberg Lab in Würzburg in 2006 I've seen an unpublished tracker for a Drosophila aggression assay that quantified aggression behaviour within a rectangular chamber in terms of top-down defined lunge counts; the tracker was written in LabView and required one of the two male flies to be painted with a white dot on the back. This tracker was later used to quantify lunges in [29].

During the last year of the project a number of trackers were published, in particular the work of [30], which addresses Drosophila aggression and courtship behaviour, has similar aspects to this work. Further, the work of [**?**] quantifies behaviour of large number of flies and [32] comes with a multiple camera 3D tracking for very large half-sphere shaped arenas.

The work introduced within this thesis was developed independently from related work, however, the top-down defined classifiers in section 5.2.5 were defined after the top-down classifiers in [30] were studied, the footnotes in this section mark an attribute and a classifier condition that were inspired by [30]. All further similarities, like choosing the Hungarian algorithm for identity assignment in non-occluded sequences or circular arena detection by the hough transform are purely coincidental.

2 System Overview

The system architecture is designed to operate with several *modules*, namely the preprocessor, the tracker, the postprocessor and the annotationTool. The idea of separation is to provide exchangeable architecture parts for the system's main tasks. Modularization further enables to derive *spin-off variants* of the system as long as modified parts fit into the given interfaces. The introduced system is designed to capture Drosophila courtship behaviour.

The workflow within the modules is straightforward, information flows from preprocessor over tracker to the postprocessor module (see figure 2). The only two-way interacting component is the annotationTool, which interoperates with postprocessed data.

The overall system performs two major steps, an **image processing step** that transforms a video information into a time series representation and a **pattern recognition step** that traverses this time series data to search for biological relevant event patterns.

The preprocessor and the tracker modules cover the image processing part of the system. The tracker is the core element and the performance critical part of the system. It is applied to every individual video frame and captures its main characteristics in a time series. The preprocessor performs a bundle of one time computations that support this process. In particular the preprocessor separates the videos background (see section 3.2.1), detects arenas within that background (see section 3.2.3), splits the input video according to detected arenas[4] and checks input video quality (see section 3.2.4), figure 2 summarizes these steps within the red preprocessor shape. The preprocessor further checks a video's input format[5] and optionally initiates a transcoding step. The tracker (figure 2, yellow shape) uses the precomputed background model to extract the videos foreground. Within this foreground a body and a wing region are segmented (see 4.2.1), a unification of these two morphological components characterizes a fly (see 4.2.3).

The image processing part further subsumes several data transformation and data cleaning steps while the data is still in its image representation (see sections 3.2.2, 3.2.4, 4.2.1, 4.2.2 and 4.2.3). It thus boosts quality and plausibility of image data before the time series is extracted. Further, the preprocessor rejects videos in case minimum quality standards are not met and the video data is detected to be inappropriate for downstream computation steps. The system aims to reject videos as early as possible, especially before the time critical tracking part is executed, in order to avoid unnecessary calculation and to safe computation time.

[4] The input video is split into a number of smaller videos, each containing video information for exactly one arena.

[5] The system accepts essentially all video formats and is limited by the ffmpeg video library only, accepted video formats include .e.g. .avi, .mpeg or .vob. The .mts format delivered by some cameras requires an additional transcoding step into .avi format, but are (still automatically) processable.

The postprocessor module covers the pattern recognition part of the system. It uses a videos time series representation as information basis to derive advanced, biologically relevant characterizations of scenes. In particular (see figure 2, green shape) the postprocessor performs a final quality control, derives advanced attributes, solves occlusions (see section 5.2.2), interpolate attributes (see section 5.2.4) and calculate scores (see section 5.2.5). In order to be self-contained the postprocessor checks once more whether its input time series is plausible and consistent to given constraints[6]. The advanced attributes compose a normalized and therefore inter-video comparable time series and include the identification of the flies through the whole video (see sections 5.2.1 and 5.2.2), determination of the flies head (section 5.2.3.3) and a calculation of downstream attributes that characterize biologically relevant events (see section 5.2.5). The video consists of sequences where the two flies are occluded, i.e. overlap each other, when solving occlusions a plausible identity mapping between neigbouring non-occluded sequences is derived (see section 5.2.2), the time series data for occluded sequences or otherwise missing frames is interpolated from observed neighboring data. Finally biologically relevant event patterns are recognized and protocolled in ethograms and excel sheets.

Besides the main modules of the system comes with several supporting tools (see section 6). The annotationTool (see figure 2, blue shape) interacts with postprocessed data. It allows to inspect the advanced time series data and manually overrule the machine decisions. Such manually tweaked postprocessing data has to be re-postprocessed in order incorporate given manual information, such that corresponding data pieces but also directly and transitively dependent data are updated accordingly. This ensures consistent data views and avoids delaying recalculations during data annotation.

The system's further comes with a web interface (see section 6.1 p. 158) which allows to process and handle large numbers of videos in a convenient way. The web interface allows to upload videos and interoperates with the system's modules. In particular, it sends the processing tasks to a computer cluster and provides processing results online. It further allows to invoke the annotationTool in order to inspect data on a local machine and supports reprocessing of annotated parts in the computer cluster. The web interface has been integrated as a control center for the overall system, it therefore has interacts to all system components and allows to monitor status and progress of computations and summarizes logging information, intermediate and final results per video.

The overall system is designed to minimize required user interactions. The system is therefore designed to be self-contained and automatically detects necessary parameter settings directly from the video material. In particular it

[6] The postprocessor could rely on guarantees that come with the current tracker implementation, however, it performs another quality check in order to check the input time series data, probably coming from a different tracker without those guarantees. This is particularly useful during development phases when operating with different versions of trackers or with spin-off variants.

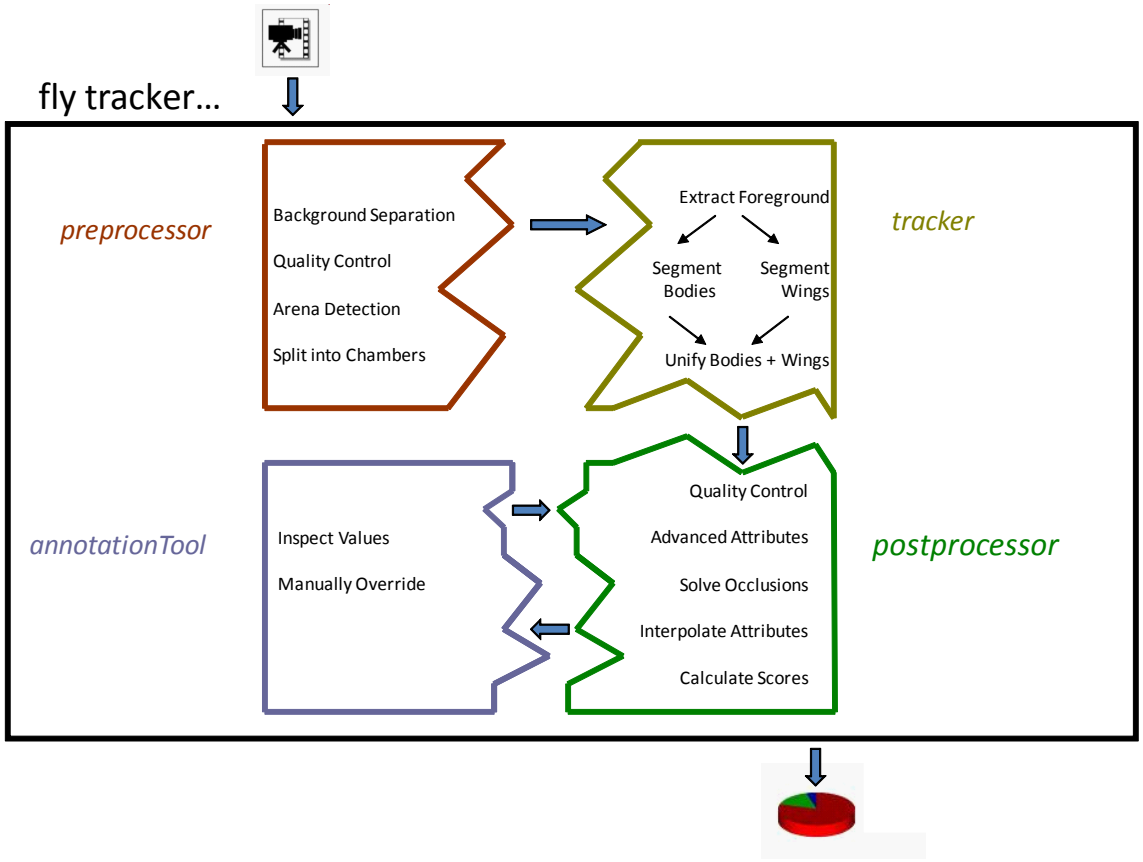


Figure 2: The black box contains the main system components (depicted by colored polygons), major workflows (depicted by the arrows) and component interfaces (depicted by matching polygon shapes). Each component (preprocessor, tracker, postprocessor, annotationTool) consists of a bundle of component specific methods. The symbols outside the black box resemble the system interfaces, the system takes videos as input and generates a summary of biologically relevant information.

is **not necessary to manually tweak parameters** or system settings e.g. for image processing purposes[7]. The only required input - taken once for *all* videos recorded in the same biological setup - is the arena size [in mm] for that setup[8], this information enables to normalize measured pixel distances and areas accordingly. The system automatizes all attention critical parts, it checks video quality standards, detects arenas and automatically computes the video background[9], the thresholds for bodies and wings and an identity mapping for occlusions[10].

Without need for manual parameter settings or calibrations the system comes in a ready-to-use state and avoids errors due to wrongly set parameters or variations in the parameter setting. The system still provides the possibility to manually set parameters, in particular there are alternative methods and implementations available for essentially all computational tasks; choosing such an alternative method may improve processing of certain assays or video classes. However, the methods described in this thesis compose a functional snapshot that is specialized for standard videos and courtship behaviour analysis.

The modular system architecture further implicitly enables to exchange modules as a whole and therefore further supports generation of spin-off applications. This may be particularly important in case the recording setup or the underlying biological experiment changes or requires adaption, or in case other behaviours of interest should be analyzed. The architecture itself supports multiple experimental setups, sections 6.3.1 and 6.3.2 provides an impression how such spin-off applications may look like.

The further sections describe workflow and methods for each of the main system components. The following section 3 covers the preprocessor which mainly is splitting the video into smaller videos each containing just one arena, section 4 then describes the tracker, the core element of the system which is extracting time series data and section 5 introduces the postprocessor which detects biological relevant data within an extended time series.

[7] When the project was started the Noldus® software bundle used similar image processing methods but required to manually mark arena regions and to manually set thresholds for foreground segmentation. Such interactive parametrization has its advantages but is inappropriate for a high throughput scenario.

[8] Standard courtship chambers in used in the lab have a diameter of 10 mm, the system uses this as a default value.

[9] The video background is software-approximated in a sufficient way (see section 6), therefore there is no need to e.g. initially record the setup with empty chambers.

[10] The system provides the option to manually overrule the machine identity assignment (see annotationTool, p. 6.2). However, the machine assignment is again designed to minimize user interaction by its self-correcting property (see section 5.2.2), due to its low error rate (on average less than a second in a 10-minute video) it requires little or no attention.

3 Preprocessor

The preprocessor, in a nutshell, reads the original video, extracts a background image and foreground video, detects individual arenas, and splits the video into smaller videos each containing a single arena. The preprocessor prepares the video for tracking for almost all existing video formats. Accepted video formats are all formats contained in the ffmpeg video library (www.ffmpeg.org), which includes all common video formats such as .avi, .mpg, .m2s and .vob. The .mts format coming from high-definition cameras requires an additional conversion step before preprocessing.

The first image processing step is to compute a video's *background*, this is done by taking the pixelwise median of several frames. The background contains *immobile* video information, within the background static fly arenas (see section 3.2.3, arena detection) may be detected.

Another major purpose of having a background is to subtract it from the original video, which derives the videos *foreground*. The foreground contains *mobile* video information, within the foreground the flies may be detected (see section 4.2.2).

Further, the background may be used for quality control purposes. It is required that the arena plate is not moved during a video as this would affect the extracted background models, such that a successful downstream analysis (like arena detection within that background) cannot be guaranteed. Further intrusion of fly arenas is disallowed, as this would disturb foreground detection, long lasting intrusions may affect background extraction as well. Analysis of the background model and its underlying frames aim to provide early (see section 3.2.4) quality control.

Section 3.2.1 describes the extraction of different background models, a quality boosting step called illumination and color correction in section 3.2.2 further refines these backgrounds for quality control and segmentation purposes. After the arena detection step in section 3.2.3 information about exact number, position and size of arenas is available, the detected size information is later used to compensate different camera zooms and essential for comparison of different videos. Before invoking individual steps of the detection sequence several quality control checks, section 3.2.4, are designed to automatically sort out videos that are not conform to defined recording standards.

Section 3.1 below describes the flow of information within the preprocessor module, in describes the *sequential order* in which the preprocessor's one-time-computation tasks are computed; Section 3.2 describes preprocessor's major functional methods in detail. In section 3.2.4 quality standards for recorded videos are discussed.

Main purpose of the preprocessor is to extract necessary background and arena information for the tracker in 4.

3.1 Workflow

As a very first step the preprocessor checks whether the video comes in a readable format and initiates a transcoding step if required. Currently all major video formats are supported, the Matlab video reader has some known issues with the .mts format which is therefore transcoded to into a .avi container. As soon as the video is guaranteed to be readable it is ready for the preprocessor's image processing steps.

Figure 3 depicts the workflow of the preprocessor module.

The preprocessor starts by selecting a set D of distributed frames (for formal definition see section 1, p. 22) and extracting a median background bg_D (formal definition p. 22) from these frames. Figure 3 (a) symbolizes this **background extraction step** and depicts this naive median background bg_D. In step (b) the frames D and the background model bg_D are **illumination and color corrected** (see 3.2.2), corrected frames and model are denoted as $D\prime$ and $bg_{D\prime}$. This step compensates varying lighting conditions and boosts video quality beyond possible recording conditions.

The corrected frames $D\prime$ enable the first quality control step (c), consisting of a **movement detection step** where undesired movements of the recorded chambers are detected and a distortion correction step that compensates undesired distorted recordings. Subfigure (c) symbolizes that step and depicts a picture from a moved and therefore rejected chamber. In general, quality control steps aim to reject videos or parts of videos that are not conform to defined quality standards (summarized in section 3.2.4) and generate pictures (like in 3 (c) or (h)) that visually underlines resp. justifies the rejection.

Withing background model $bg_{D\prime}$ the arenas are detected, subfigure (d) symbolizes the arena detection step and depicts an accumulator array of a circular hough transform. This transform may be applied to detect circular structures within an image (details in section 3.2.3), proper arenas are further restricted to not overlap each other or with the video border. The arena detection step detects number, position and size of recorded arenas, all points within proper arenas are summarized in an arena map $\mathfrak{A}$.

Having arenas, the final refinement of the background model may be derived. Figure 3 (e) and (f) symbolize two **background smoothing step**, (e) depicts the rigorously smoothed background bg_{SA} used for body segmentation (f) depicts the cautiously smoothed background bg_{sF} used for wing segmentation. For the background depicted in (e) points within arenas - the points in arena map $\mathfrak{A}$ - are smoothed, this refined background allows to **detect fly bodies**, for the background depicted in (f) flybody-like regions within an arena are smoothed (see section 3.2.1, p. 23 for more details).

Segmentation ability allows some final quality control steps. Subfigure 3 (g) symbolizes a **border intrusion check step** and depicts watched boundaries for all detected arena. These boundary region $\mathfrak{B}$ consists of ring-like structures[11],

[11] The thickness of the watched boundary rings, i.e. the outer radius minus the inner radius, is set as half of an average fly length; This fly length may be determined as soon as the

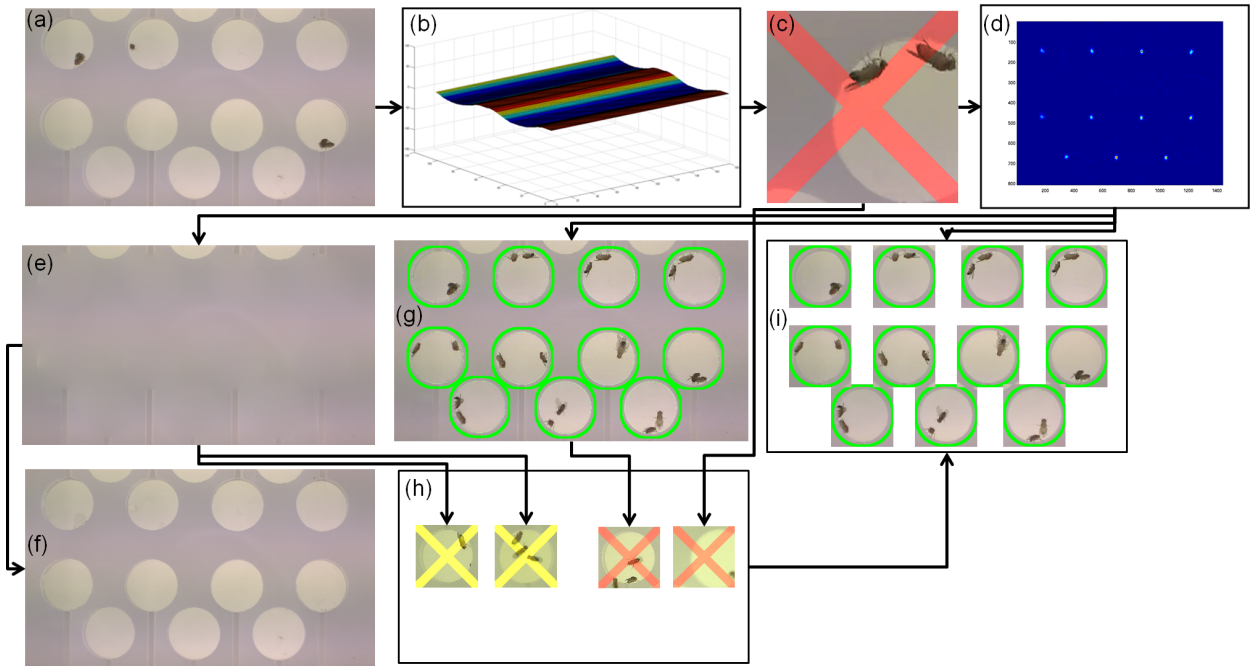


Figure 3: (a) symbolizes the background extraction step and depicts the naive median background (b) symbolizes the illumination and color correction step and depicts the illumination correction curve (c) symbolizes the movement detection step and depicts a picture from a rejected, moved chamber (d) symbolizes the arena detection step and depicts the accumulated peaks for circular arena detection (e) and (f) symbolize the background smoothing step, (e) depicts the rigorously smoothed background used for body segmentation (f) depicts the cautiously smoothed background used for wing segmentation (g) symbolizes the border intrusion check step and depicts the watched boundary for each detected arena (h) symbolizes the summarized quality control steps by depicting reasons for rejecting chambers, which are (left to right) too few flies per chamber, too many flies per chamber, chamber boundary intruded by object from outside, chamber was moved (i) symbolizes the arena splitting step and depicts the individual chambers and their watched boundaries without further background in between. The black arrows mark direct dependencies within the information flow, the numbering of subfigures (a)-(i) denotes the processing order.

that enclose and aim to protect the interior arena points in $\mathfrak{A}$, the boundary points in $\mathfrak{B}$ in are eagerly observed in order to detect arena intruding objects. Such objects may include fingers or hands of an unwarily recording person or objects that temporarily cover arenas or parts of arenas, in particular escaped flies from previous experiments may disturb or influence recorded behaviour data (see also section 3.2.4). The third picture in (h) depicts an intruding fly that was captured by the border intrusion check.

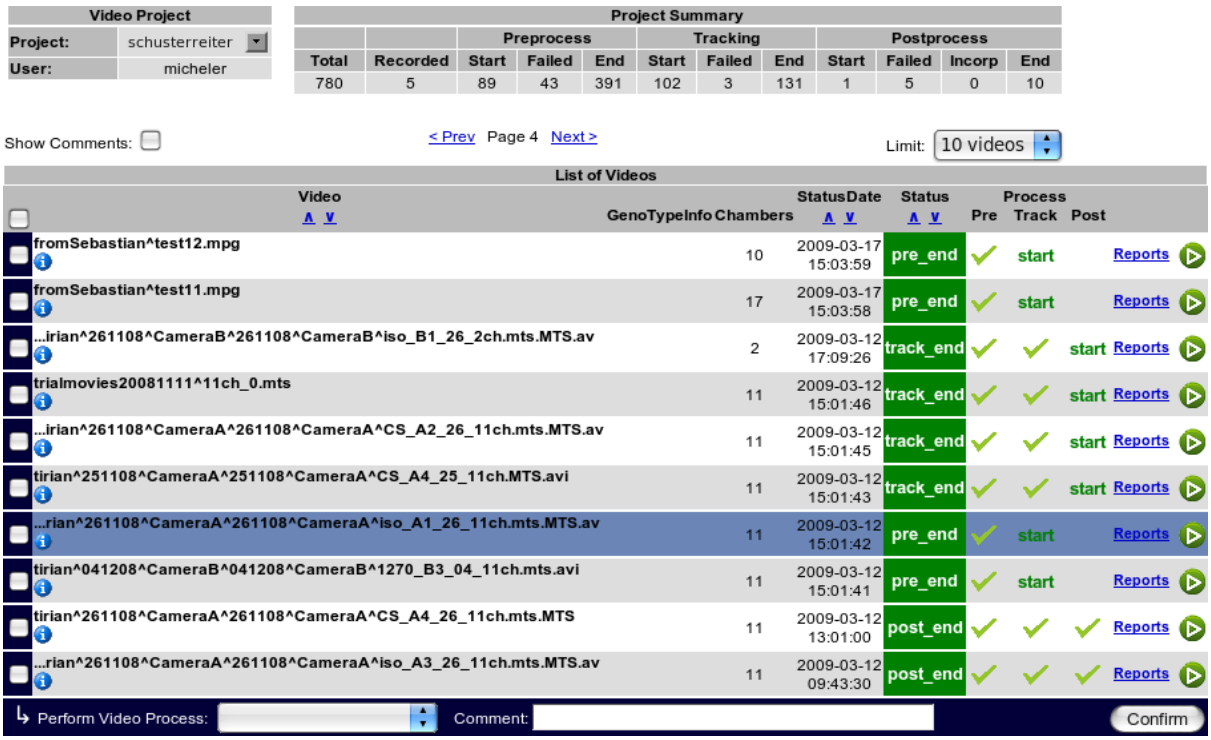
Further, segmentation allows to check the number of flies per arena, the to-be-analyzed courtship behaviour assays require arenas with exactly two flies. Subfigure (h) symbolizes the summarized **quality control steps** and depicts common reasons for rejected chambers, which are (left to right) **too few flies per chamber**, **too many flies per chamber**, **chamber boundary intruded by object from outside** and **chamber was moved**. The arrows to the pictures in (h) denote preprocessor steps that may raise particular rejection reasons.

Non-rejected arenas may be processed by downstream modules, subfigure 3 (i) symbolizes the final **arena splitting step** and depicts individual chambers and their watched boundaries without further interconnecting, but dispensable background information. The original video is split into smaller single-chamber videos that capture smallest rectangles containing an arena and its watched boundary only. Besides the video information, all further preprocessor information, like background models and arena map, are cut and separated accordingly, such that a full set of preprocessor information is generated for each single-chamber video as if each of these videos had been individually preprocessed and such that re-preprocessing these single-chamber videos would not provide further information.

Figures 4 and 5 show screenshots of the webinterface and depicts the information summary for a preprocessed video. Figure 4 shows a list of processed videos, the videos in status *pre_ end* successfully finished the preprocessing step and are ready for further processing. When clicking the icon below the video name information about individual video chamber is shown as in figure 5. The preprocessor generates the two pictures on the left and a acception or rejection picture per chamber. The figures on the left depict the first frame of the video overlayed with the watched boundary, this picture also gives an impression about the arena detection results, further a legend depicting arena numbers - numbers are placed at corresponding arena positions in the picture above - is provided to support the user to correctly map processing results to his input recordings. A list of single video chambers is provided with a per-chamber processing status and an acception or rejection picture, specifying whether a particular chamber passed all quality control steps. The preprocessing results of the sample video in figure 5 reject chamber 5 since it is empty and accepts all other video chambers.

The major milestone within the preprocessor workflow is the arena detection step depicted in figure 3 (d), it enables all further processing steps, in particular

smoothed background model and therefore body segmentation is available.



Video Project	
Project:	schusterreiter
User:	micheler

Project Summary											
		Preprocess			Tracking			Postprocess			
Total	Recorded	Start	Failed	End	Start	Failed	End	Start	Failed	Incorp	End
780	5	89	43	391	102	3	131	1	5	0	10

Show Comments: < Prev Page 4 Next > Limit: 10 videos

List of Videos

Video	GenoTypeInfo	Chambers	StatusDate	Status	Pre	Process Track	Post	
fromSebastian^test12.mpg		10	2009-03-17 15:03:59	pre_end	✓	start		Reports
fromSebastian^test11.mpg		17	2009-03-17 15:03:58	pre_end	✓	start		Reports
...irian^261108^CameraB^261108^CameraB^iso_B1_26_2ch.mts.MTS.av		2	2009-03-12 17:09:26	track_end	✓	✓	start	Reports
trialmovies20081111^11ch_0.mts		11	2009-03-12 15:01:46	track_end	✓	✓	start	Reports
...irian^261108^CameraA^261108^CameraA^CS_A2_26_11ch.mts.MTS.av		11	2009-03-12 15:01:45	track_end	✓	✓	start	Reports
tirian^251108^CameraA^251108^CameraA^CS_A4_25_11ch.MTS.avi		11	2009-03-12 15:01:43	track_end	✓	✓	start	Reports
...rian^261108^CameraA^261108^CameraA^iso_A1_26_11ch.mts.MTS.av		11	2009-03-12 15:01:42	pre_end	✓	start		Reports
tirian^041208^CameraB^041208^CameraB^1270_B3_04_11ch.mts.avi		11	2009-03-12 15:01:41	pre_end	✓	start		Reports
tirian^261108^CameraA^261108^CameraA^CS_A4_26_11ch.mts.MTS		11	2009-03-12 13:01:00	post_end	✓	✓	✓	Reports
...rian^261108^CameraA^261108^CameraA^iso_A3_26_11ch.mts.MTS.av		11	2009-03-12 09:43:30	post_end	✓	✓	✓	Reports

Perform Video Process: Comment: Confirm

Figure 4: Screenshot of the webInterface showing a list of videos in different processing stati. A click to the *start* field initiates the next processing steps, a click to the *Reports* link provides details for the processing so far.

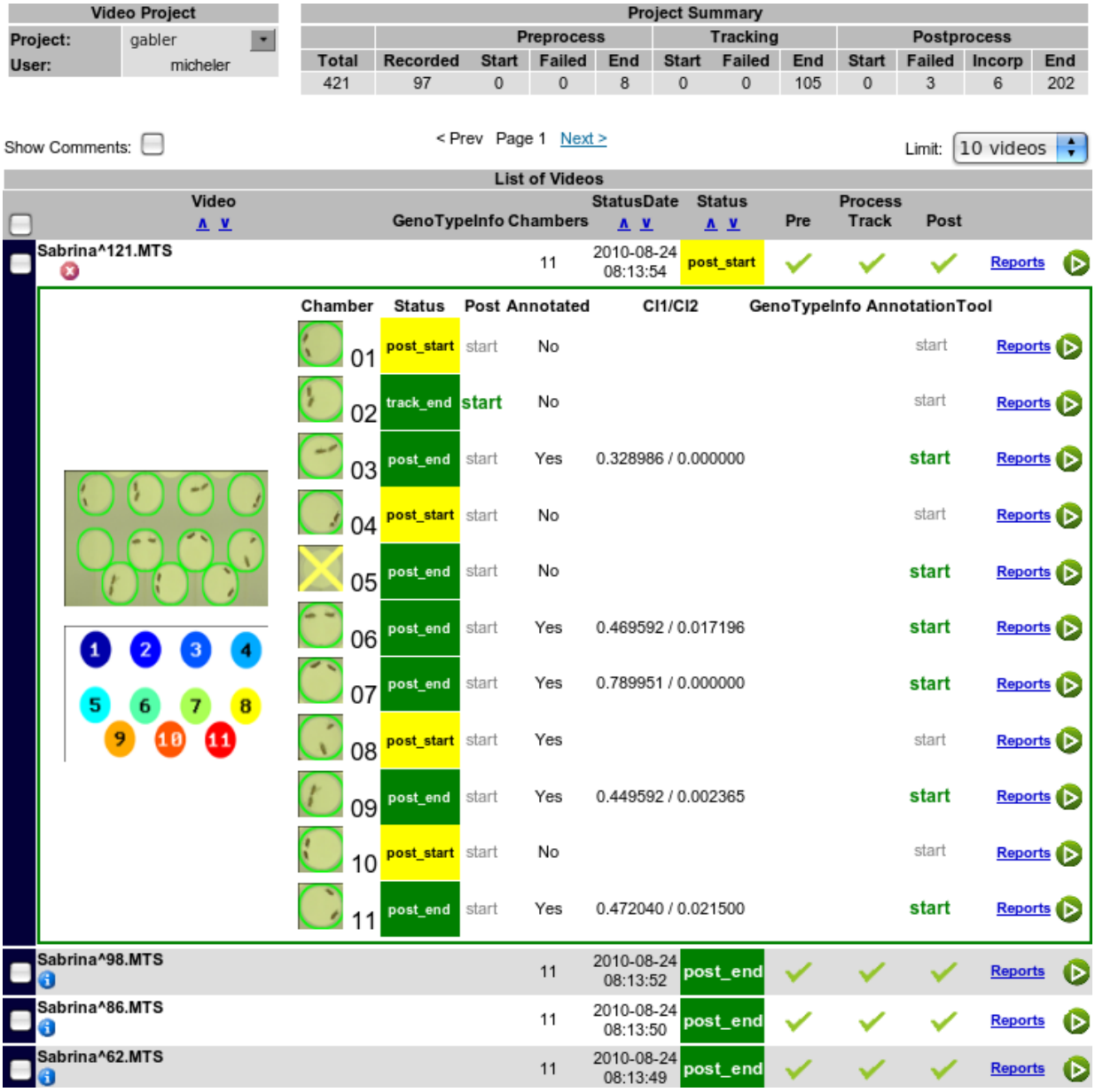


Figure 5: Screenshot of the webInterface showing per-chamber details. Two summary pictures on the left depict the watched boundary and chamber numbers, each chamber row starts with a quality control acception or rejection picture. Chambers with in a yellow state are currently being processed, green states indicate that processing is done. Clicking a green *start* field in column AnnotationTool invokes the annotationTool with video and tracking information for the selected chamber, already annotated chambers are marked with a *Yes* in column Annotated.

the steps depicted in 3 (e), 3 (g) and 3 (i) are directly, 3 (f) and 3 (h) transitively dependent on arena detection. Background smoothing in 3 (e) enables fly body segmentation and therefore allows to derive the background 3 (f) and to check correct number of flies. Arena detection (d) requires that its input background model comes from a video that was neither moved nor distorted, step 3 (c) ensures that, and movement detection 3 (c) is majorly improved when operating on illumination corrected backgrounds and video frames, which are provided by step 3 (b).

The information used by the tracker module are mainly the results of steps 3 (e), 3 (f) and 3 (i). The first two are used for segmentation purposes, figure 28 in section 4.2.1.3 (p. 61) depicts the segmentation workflow with the results of 3 (e) and 3 (f) as required input. The tracker module processes each of the quality controlled single chamber videos, generated in step 3 (i).

The method section 3.2 provides background information and details for the preprocessor's major image processing steps.

3.2 Methods

3.2.1 Background Extraction

This section describes the extraction of different background models, which are used for segmentation, arena detection and quality control purposes.

The *median background model* bg_D is the first computed and simplest background model, it is defined as a pixelwise median of distributed video frames (more detailed and formal definitions below). All further backgrounds in this section are refinements of this basic model and base - directly or transitively - on its values.

The intuition behind the median-based definition of bg_D definition is that it captures immobile background information under the assumption that relatively small and mobile foreground - i.e. the flies - indeed *moves*, and therefore unveils every background pixel more often than it conceals it. Although this is obviously not always guaranteed, bg_D is still mostly sufficient for all downstream tracking purposes. However, different refinements of the background model help to simplify and improve forthcoming image processing tasks. Obviously, too little moving flies may be eliminated from the background either by taking a higher pixelwise quantile, or by smoothing out undesired foreground information.

The first refining step is an illumination and color correction, section 3.2.2 describes how a corrected background $bg_{D'}$ is derived from the basic background bg_D. The corrected background enables to boost videos quality, it improves segmentation (see 4.2.1) and is essential for quality control (see 3.2.4) as it dramatically reduces false alerts. Illumination correction allows processing of videos that would otherwise not pass the quality control tests, in particular the

checks whether the video was moved and the border intrusion check rely on the corrected picture sets and background model.

Intuitively, the illumination and color corrected background $bg_{D'}$ is used to compensate non-static and uneven lighting conditions, it allows to increase the quality of both, the background model and its input video frames, beyond possible recording conditions.

The corrected background image is further smoothed by a roifill operation, which "fills in a specified region of interest polygon in a grayscale image. roifill smoothly interpolates inward from the pixel values on the boundary of the polygon (...) and can be used (...) to erase objects in an image" [6]. The smoothed background is used for segmentation.

Intuitively, roifill is used to eliminate immobile flies or fly parts from the background, such that - after background subtraction - the foreground contains the flies.

Formally, the basic background model bg_D is calculated as the median intensity for each pixel across a series of 100 frames spaced evenly across the video.

In particular, the set D of equally distributed frames and the pixelwise median background bg_D are defined as follows:

Definition 1 *Let l be the length of a video v, and d be the number of frames to be considered, and f_i the ith frame of video v. The set D of equally distributed frames is then defined as $D = \{f_i | i = \Delta f \cdot k, \Delta f = \frac{l}{d-1}, k = \{1..d\}\}$.*

Definition 2 *Let $\mathfrak{d}$ be a set frames and $f^{(x,y)}$ the value of the pixel positioned at (x, y) in frame f. The pixelwise median background $bg_\mathfrak{d}$ is then defined as $bg_\mathfrak{d}^{(x,y)} = median_{f \in \mathfrak{d}}(f^{(x,y)})$.*

The background bg_D, with $\mathfrak{d} = D$, is used to illumination- and color-correct the input video frames in D (see 3.2.2), the corrected frame set $D\prime$ is then used to recalculate a corrected background $bg_{D'}$, as the pixelwise median background with $\mathfrak{d} = D\prime$.

The median background bg_D (resp. its color corrected descendant $bg_{D\prime}$) provides a good estimation for the background of the video v, typically the background captures the immobile part of the video. In order to eliminate remaining mobile parts from the background $bg_{D'}$ a further smoothing step (see below) is performed by applying a Matlab roifill operation.

The Matlab roifill operation fills in a specified region of interest by smoothly interpolating inward from given boundary pixel values. Mathematically this is achieved by solving Laplace's equation with Dirchilet boundary conditions, such that function ψ of the Laplace equation $\nabla^2\psi = 0$ and function v defining values V_B for boundary B match within B, $\forall x \in B : v(x) = \psi(x)$, see [7],

[8] for a general and discrete solution and proof of applicability. Intuitively roifill computes values for a discretized Laplace's equation such that interpolated interior pixel values equal the average of their 4-connected neighborhood while boundary pixel values are constrained to equal the given original image.

In practice roifill is called with an original image I and a binary mask specifying a to-be-interpolated *region of interest* $\mathfrak{r}$, $\mathfrak{r} \subseteq I$, boundary B resp. values V_B of function v are derived as the *perimeter*[12] of $\mathfrak{r}$ resp. its values in I.

For segmentation two smoothed background models are used[13], for this reason two different smoothing operations, differing in their region of interest $\mathfrak{r}$, are defined.

The more rigorous operation roifills with $\mathfrak{r} = R_A$, with R_A being a dilated[14] arena map A[15], and smoothes out entire arenas. Figure 6 (a) depicts backgrounds bg_D for video v, arenas 1, 2, 8 and 11 (see legend in the middle) contain static content caused by dirt pieces (11), immobile (1,8) or slowly moving (2) flies. This static content is removed by the rigorous smoothing operation, figure 6 (b) depicts the rigorously smoothed background model bg_{SA}, derived as $bg_{SA} = roifill(bg_{D'}, R_A)$. Obviously all static content has been eliminated from the background, and may therefore be detected as potential foreground regions.

The more cautious operation roifills with $\mathfrak{r} = R_F$ and smoothes out the potential foreground parts R_F only, R_F is derived by segmenting background $bg_{D\prime}$ - as if it was a video frame - with method $fast2$[16] and background bg_{SA}. Regions segmented as foreground would be detected as flies or fly parts by later segmentation steps and are therefore eliminated from $bg_{D\prime}$, such that they remain within the foreground after background subtraction. Figure 6 (c) depicts the cautiously smoothed background model bg_{sF}, derived as $bg_{sF} = roifill(bg, R_F)$. By directly comparing subfigures (b) and (c) some differences may be noticeable: While (b) radically eliminates all foreground - even the arenas themselves - (c) still contains fine-grained details of the background. While the flies or fly parts in arenas 1, 2 and 8 are eliminated[17], dirt pieces like in arena 11 remain

[12] The perimeter of a binary mask R may easily be derived by subtracting its morphological erosion R^- from the original mask, $perimeter = R - R^-$.

[13] During segmentation two foreground regions (a body region and a body plus wing region) are extracted, both segmentation steps use differently smoothed background models.

[14] Figure 16 depicts watched boundary rings around arenas, for smoothing purposes R_A is the inner area of such a ring, which is computed by $\lfloor \frac{c}{2} \rfloor$ repeated morphological dilations of arena map A (see section 3.2.3), where c is a default crop offset value (used to crop individual single chamber videos out of recorded multi-chamber videos).

[15] Arena map A masks all detected arenas within a video, i.e. the regions where the flies are in; c.f. chapter 3.2.3 (arena detection), for definition and detection of arenas.

[16] C.f. section 4.2.1 for more details and definition about method $fast2$.

[17] One may notice that very slight shadows remain where the flies have been. This results from blurred flies within the median background, where segmented foreground region R_F is surrounded by a blurred region between fly and arena brightness. Although R_F is dilated in order to deal with surrounding effects, a slight artefact shadow may remain after interpolation. However, this effect and does not affect segmentation quality, as long as the shadowed region covers less than 50 % of the arena, which is reasonable to assume and given for all used arena sizes.

in the background, and therefore vanish later from the foreground.

The rigorously smoothed background bg_{SA} is later used to segment fly body regions, the cautiously smoothed background bg_{sF} is used to segment body plus wing regions. While a fly body regions provide a good contrast to the arena background, the much brighter and volatile wing regions are more prone for miscaptures of dirt regions. For this reason both background models are motivated, one that guarantees to eliminate all disturbing content while the other captures fine-grained background information, and therefore helps dealing with undesired static patterns.

Figure 6 provides a gallery of different background models for a sample video, in (a) the original median background bg_D is shown. Unfortunately it contains dark static content in chambers 1, 2 and 8 and a bit of bright static content in chamber 11. A major question is: How to eliminate such foreground parts from the background?

One approach is to use - instead of the median background - a higher quantile background, which may tolerate less moving flies. Still it would leave the question where to set the threshold. Too bright background would affect the segmentation by producing foreground artefacts. In tricky cases, like absolutely static anesthetized flies, the background is simply not unveiled, and no quantile - not even the maximum - would help to derive a "clean" background (see below and (d)). However, the alternate approach - smoothing - works independently from fly movement. Figure 6 (b) shows the rigorously smoothed background bg_{SA} where arenas are smoothed, this rigorous smoothing approach guarantees to eliminate all static content from the background, figure 6 (c) shows the cautiously smoothed background bg_{SF}, where static content that would later be segmented as flies are smoothed. The cautious smoothing approach eliminates the obvious dark static content in chambers 1, 2 and 8, but does not affect the bright static content in chamber 11. Finally, figure 6 (d) shows the pixelwise maximum values, this background model was computed for comparison purposes only. Obviously, the static parts in arenas 2 and 11 are eliminated by picking a higher - the highest possible - quantile threshold. However, static content in arenas 1 and 8 are reduced, but still visible.

The observation in (d) underlines that picking quantile thresholds higher than the median may reduce static content effects, but do not guarantee to eliminate the problem. Therefore the smoothing approach was chosen, the smoothed backgrounds in (b) and (c) are used for segmentation purposes.

However, in case the background contains static obstacles, for example a food source or simply pieces of dirt accumulating over time, these are smoothed and therefore "moved" to foreground as well. In case of presence of e.g. a food source, i.e. a given static object within the chamber, this may have undesired effects, and taking a higher pixelwise quantile (or even the maximum) is easier to handle (see chapter spin-offs, section 6.3.2). The presence of dirt is handled separately and turned out as a challenge for itself. Two different types of dirt pieces may be encountered, dark-colored pieces potentially appearing as

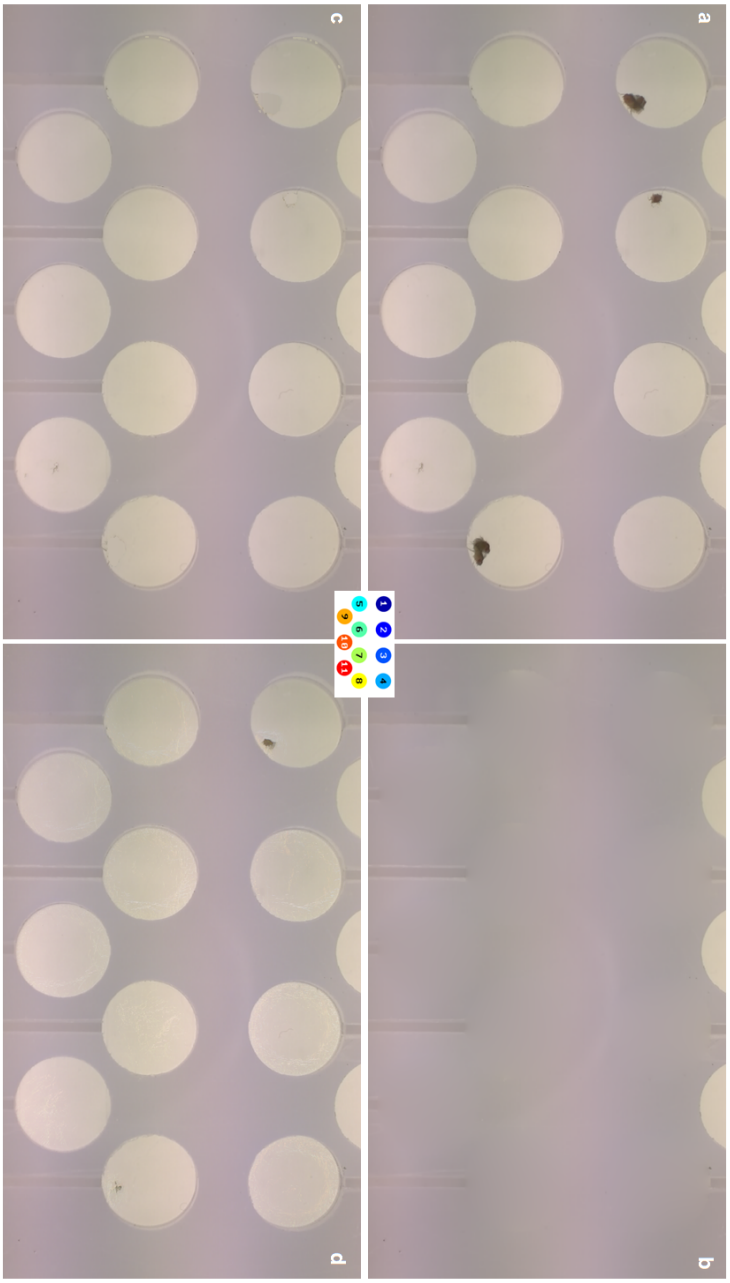


Figure 6: Gallery of Background Models. (a) median background bg_D. (b) rigorously smoothed background bg_{SA}, used for fly body segmentation. (c) cautiously smoothed background bg_{sF}, used for fly wing segmentation. (d) Pixelwise maximum of frames in $D\prime$. (middle) Visualization of arena numbering, left to right then top to down.

foreground and light-colored pieces on the top glass, potentially appearing as background and therefore potentially separating fly regions into smaller pieces. While the tracker deals with dirt pieces in order to ensure proper fly regions (sections 4.2.2 and 4.2.3), the postprocessor still checks for static objects and deals them separately resp. merges fly attributes in case of an unexpectedly encountered body or wing regions.

The purpose of background extraction is to provide a background model for segmentation and arena detection purposes, the purpose of the particular smoothing steps is to improve background quality for segmentation purposes. Segmentation operates on *foreground* images (see chapter 4.2.1, in particular figure 23), which are derived by subtracting the derived background from the original video frames.

3.2.2 Illumination and Color Correction

For each video input frame two different corrections are done: Illumination correction is about compensation of uneven lighting conditions, color correction is about adjusting colors in different frames to a common level.

Sometimes the border of video frames are darker than the center or troublesome camera settings result in disturbing effects in the picture. In case of e.g. low shutter speed settings dark or bright bars are present, and since these bars are non-static - they are moving very slowly in y-direction - they are not guaranteed to be compensated by the background model. Rather than computing different segmentation thresholds for specific frame regions undisturbed, even-illuminated images are recomputed from the disturbed images and a known background model.

Intuitively the different pictures are used to "correct each other", such that the background represents "what they all have in common" and each picture compensates its differences the background.

In order to compensate illumination a disturbance field is subtracted from video input frames, in order to compute that field (1) the (also "uneven") fly and arena information have to be removed (see figure 7 (a)-(c)). A simple arena detection per brightness is used, which is less accurate than arena detection by hough transform (see arena detection below), but sufficient for reasonable median estimation. Arena detection per hough transform relies on a quality controlled background model and is therefore not yet available.

Then (2) the morphological closing of the linewise median of the remaining values – a disturbance vector - is computed and (3) the disturbance field is derived by reconstruction of the original image size, which is constructed by columnwise repetitions of that disturbance vector.

To overcome color differences within different illumination corrected frames their color is adjusted to the color levels of their (shared) background model. Therefore, per color layer, the medians of all fly-and-arena-removed values are

computed for the background and illumination corrected frames and their differences are subtracted from the corresponding illumination corrected frames (see algorithm below).

For segmentation purposes - a performance critical part of the software - background-subtracted images are illumination corrected by a simpler and faster method. Since arenas are already detected at that point arena information is used to remove non-arena values, then the linewise median of a background subtracted image is computed as the disturbance vector, again the disturbance field is reconstructed by columnwise repetitions and finally that field is subtracted from the background subtracted image.

Illumination correction and color correction improve video quality beyond best possible recording conditions and keep thresholded segmentation simple. A illumination and color correction algorithm was introduced to correct input video frames and according to their median background model, and a simplified illumination correction algorithm to correct input video frames according to a corrected background model and given arena information.

In particular, illumination correction and color correction are used to compute a corrected background model - by computing a background model bg_D of disturbed pictures D, then illumination-correct and color-correct these pictures in D to $D\prime$ according to their background model bg_D[18]. The corrected picture set $D\prime$ is then used to compute a *corrected* background $bg_{D'}$ and to perform the quality control checks, in particular the checks whether the video was moved and the border intrusion check rely on a corrected picture sets and background model.

The pictures in figure 7 show the illumination distribution of an example pic (a) from the top left view, (b) from top, (c) from side and (d) a disturbance curve representing the uneven illumination. In order to compensate illumination this curve is subtracted from video input frames, in order to compute the curve the (also "uneven") fly and arena information has to be removed (see 7 (b)).

Algorithm 3 *Illumination and color correction*

subtract the video input frame F from the background model bg_D
if arena information given, eliminate color values within arenas
otherwise, for each color layer
    eliminate color values of flies
    detect arenas by brightness, and eliminate its color values

compute column vector v_C as morphological closing of the linewise median of the remaining values

[18] intuitively, the different pictures in D are used to "correct each other", such that bg_D is "what they all have in common" and each picture compensates its differences to bg_D

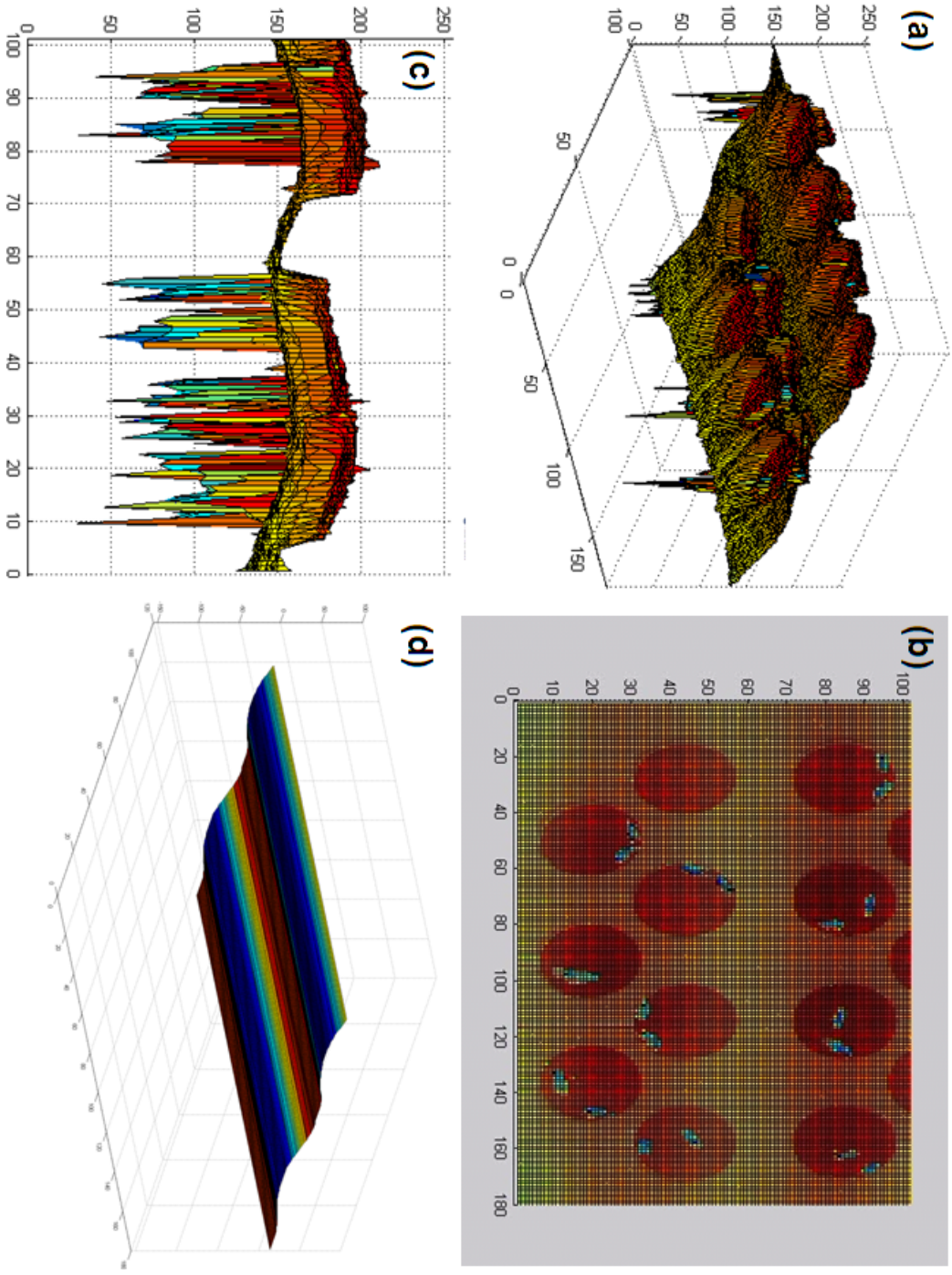


Figure 7: (a-c) Heatmap of gray values from different perspectives. (a) from top left view (b) view from top (c) view from side (d) resulting compensation curve.

and derive disturbance curve as a reconstruction of the original image size by columnwise repetitions of v_C.

add disturbance curve to the original video input frame to derive illumination corrected frame F'

for each color layer

compute the medians of and bg_D and F'

subtract their difference from each color value in F' to derive F''.

At the beginning fly and arena information is removed, a simple arena detection per brightness[19] is sufficient for reasonable median estimation. The frame F' is the illumination corrected frame, F'' is the illumination and color corrected frame.

For segmentation purposes - a performance critical part of the software - background-subtracted images are illumination corrected by a simpler and faster method.

Algorithm 4 *illumination correction in background-subtracted images*

disturbedImage = background - originalImage

disturbance = linewise-median(disturbedImage)

correctedImage = disturbedImage - disturbance

First the linewise median vector in disturbed image is computed, then the disturbance is defined as columnwise repeated median vectors, up to the size of the (original) disturbedImage and finally the correctedImage is computed by subtracting the disturbance from the disturbedImage.

Figure 8 below depicts the effect of illumination and color correction on a sample video frame. In (a) the original video frame is depicted, there are clear illumination differences visible, particularly two regions in the middle appear brighter than e.g. the region at the top. The same frame, after a illumination and color correction step, is depicted in (b), the illumination and colors are visibly more homogeneous.

Illumination correction and color correction improves video quality beyond best possible recording conditions and keep thresholded segmentation simple.

[19] Arena detection by brightness is less accurate than arena detection by hough transform (see section 3.2.3), but sufficient for achieve the goal here. Note, that arena detection per hough transform relies on a quality controlled background model, which is not yet available here.

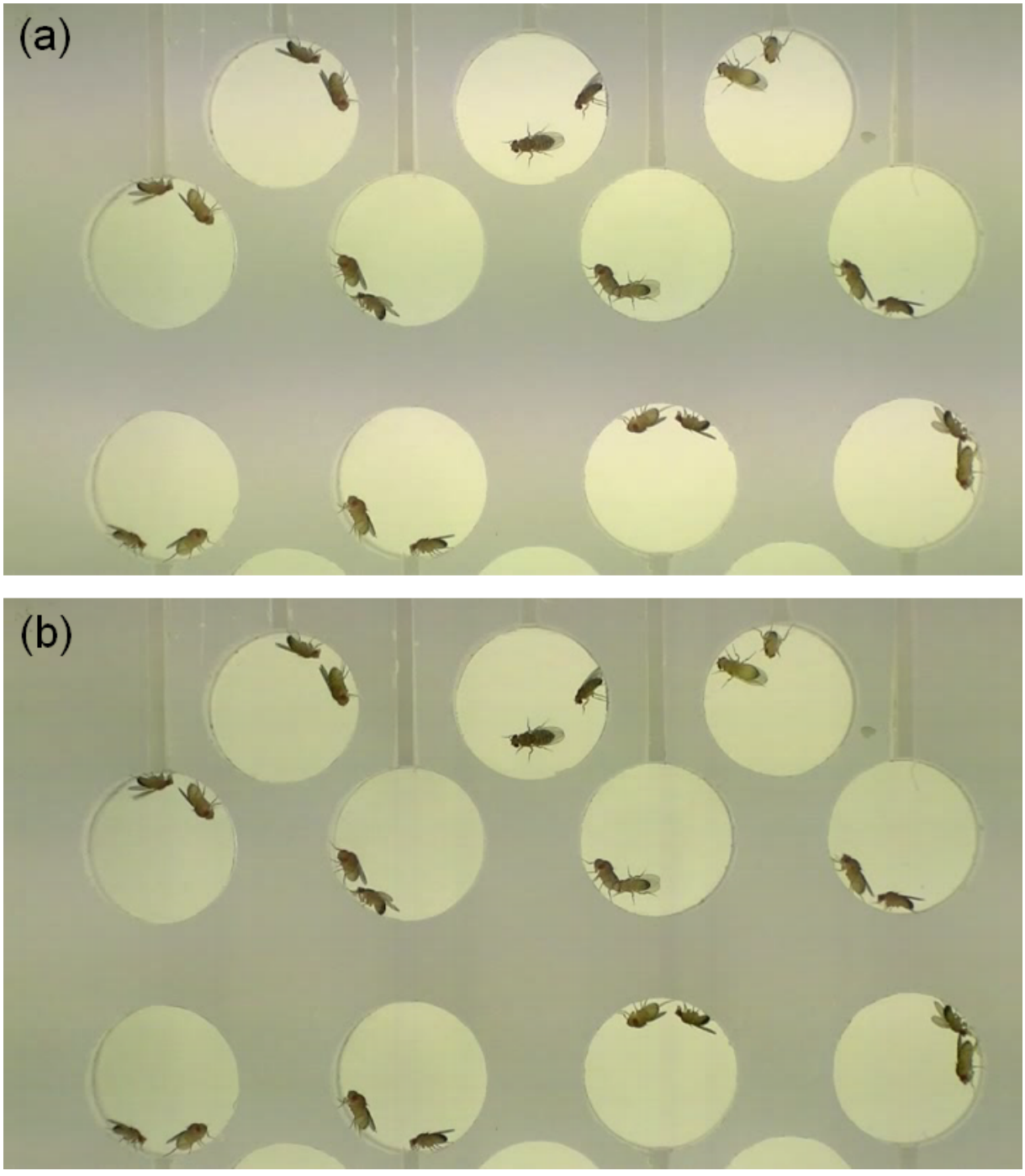


Figure 8: Result of illumination and color correction. (a) original video frame. (b) illumination and color corrected video frame.

3.2.3 Arena Detection

Automatic arena detection is essential for the system. A manual specification of arena areas would be time-consuming and against the idea of a high throughput system, and would further be prone to human errors. Especially for arena detection this is highly undesirable, since detected arena information is incorporated for many different, partly very important software parts. Most importantly detected arenas are used for normalization purposes, which means that all distances and areas, initially measured within an image in pixels, may be transformed into millimeters or square millimeters.

Recorded arenas are used as *anchor* objects, they are standardized in their size and reused for multiple video recordings. Arenas therefore come with an a-priori knowledge about their real size in millimeters. According to this and automatically detected arena radii (in pixels) the system may compensate different camera zooms and *ground* resp. **normalize** distances and other characteristics. For this reason data from individual arenas are **comparable** although arenas may still be recorded in **different videos** (see also section 5.1, p. 73).

The core element for arena detection is the *hough transform* [9][10], a mathematical operation that allows to detect instances of certain objects, like lines or circles[20]. It was originally applied to binarized gradient images, i.e. to the edges of an images, and transforms image space to a *parameter space* called hough space. When searching for circles within an image, the hough space is 3-dimensional, the axis correspond to the parameters, two midpoint coordinates and the radius, that define circles. Searching for lines transforms into a 2-dimensional hough space, figure 9 below depicts such a transformation and illustrates the principle of the hough transform.

Each point in parameter space corresponds to a line in image space. Objects are detected in parameter space are detected by a *voting procedure*. Each point in parameter space gets a number of votes from the picture in image space that corresponds to the number of pixels that are on that particular object. With image space being the binarized gradient image, which captures the edges of a picture, the parameters describing lines within the original picture get most votes. Local maxima in hough space are called hough peaks, the height of the peak is the evidence for an object with parameters determined by the peak position. The hough transform may therefore detect *imperfect instances* of certain objects. It is further possible to filter for objects with similar evidence.

Figure 9 depicts a simple original picture showing two lines and its corresponding hough transform. Lines may be described by giving a distance to the origin r and angle θ, points in hough space with coordinates (r, θ) therefore correspond to lines in image space. Within hough space, sometimes also called *accumulator* apace, then votes are accumulated, their values are depicted as gray values corresponding to their total number of votes per hough space point. The hough space in 9 (right) shows two clear hough peaks, each indicating

[20] The *Generalized* Hough Transform allows to detect *arbitrary* objects. [11]

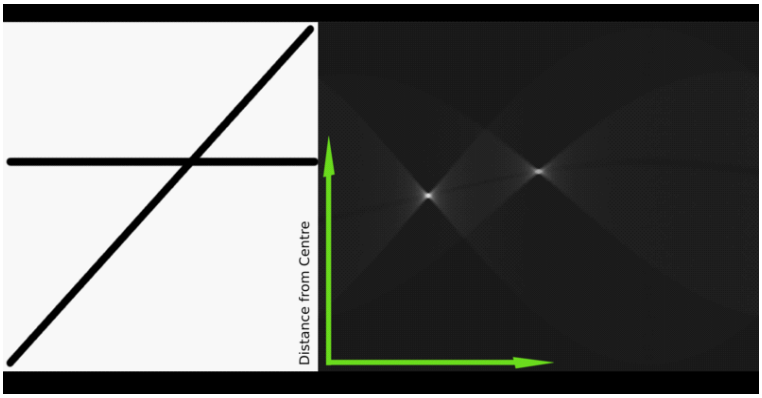


Figure 9: left: two lines in image space. right: hough transform into parameter space, the axis are r, the distance from the origin and its angle θ. source: [12]

the presence of a line in image space. The hough peak coordinates (r, θ) are parameters for the mathematical description of the two original lines.

For arena detection the *circular* hough transform is applied on the gray-scale values of the background image gradient, such that peaks in hough space correspond to circles in image space, the higher a peak, the more indication for a circle with specific centroid and radius. First centroid and radius are detected individually for all circles candidates, then - since all arenas are presumed to be of the same radius – a common radius is selected. that has both, high peaks and fits for many circles.

The main differences to the hough transform introduced from above are as follows:

- Circular hough transform: Arena detection uses the circular hough transform and therefore operates with 3-dimensional parameter space
- Gray scale gradient values: Operating with the gray scale values of the gradient image rather than the binarized gradient image allows an extension to the voting procedure, where the vote for each point is weighted by its value. While high contrast regions, showing clear edges, still have high votes, low contrast regions may still contribute, such that a large amount of weak evidence still allows to detect an object.
- Similar circle evidences : Arena detection requires that the hough peak for each circle is at least ten percent of the highest detected peak, circle candidates with lower hough peaks values are filtered. Figure 20 (c) in the quality control section 3.2.4 shows an example where circles are filtered due to a low relative peak.

From the resulting circle candidates a common radius is determined by (1) expressing the hough peak value of detected each circle as a percentage of the highest peak for that radius and (2) picking the radius with the highest sum of peak percentages as the common radius. The first step normalizes peak values in order to avoid preference of big circle radii, the second step goes for high peaks and many circles.

Arenas that are only partly visible or would overlap with other arenas are removed before radius selection. In case of circle candidate overlaps the candidate with the highest hough peak may remain. Figure 12 shows an artificial example with such potential overlaps, figure 20 (b) in the quality control section 3.2.4 shows a filtered partly visible arena.

Figure 10 (left): shows the accumulated votes in hough space. The top picture shows that the hough peaks positions correspond to circle midpoints. Bottom pictures visualizes the heights of the hough peaks which give the evidence for each circle. There are differences in relative evidences, the upper left circle has much more evidence than the lower right, however, all hough peaks are strong and clearly visible and all circles are properly detected. The right picture shows the input median background picture together with the hough-detected circles. Even fly parts in some arenas do not irritate the detection by the hough transform, its voting procedure implicitly deals with incomplete or disturbing votes.

The hough transform based arena detection algorithm is extremely robust to differences in recording setups. Figure 11 depicts median backgrounds from different videos and draws the detected arenas inside the background. In particular the background drawn in (a) contains arenas which are poorly visible arenas and extraordinary difficult to detect, even by eye. The arena detection algorithm was designed to deal with such little evidences. Figure 11 (b)-(d) depict other sample backgrounds to demonstrate the wide applicability, the backgrounds (c) and (d) even contain undesired loading bars (a piece of equipment that covers the arenas and should be removed before recording), however, arena detection still works. Figure 20 (a) on p. 46 in the quality control section depicts an example of hexagonal positioned arenas, just to demonstrate that there is no assumption about number and distribution of arenas.

Further, the algorithm aims to work in clutter, which means that encountered disturbing objects, like flies within the arenas or handwritings within background, are implicitly handled by the hough transform. However, encountered *round* objects, especially when better visible than arenas, may mislead the detection process and are therefore undesired. In particular visible screws or reflections of the camera objective have been observed to be disturbing when arenas are poorly visible as in background 11 (a).

The arena detection algorithm is extremely good in detecting circular objects, in order to demonstrate that it was applied "out of context" on a famous painting from kandinsky. Figure 12 (a) shows the original painting since there

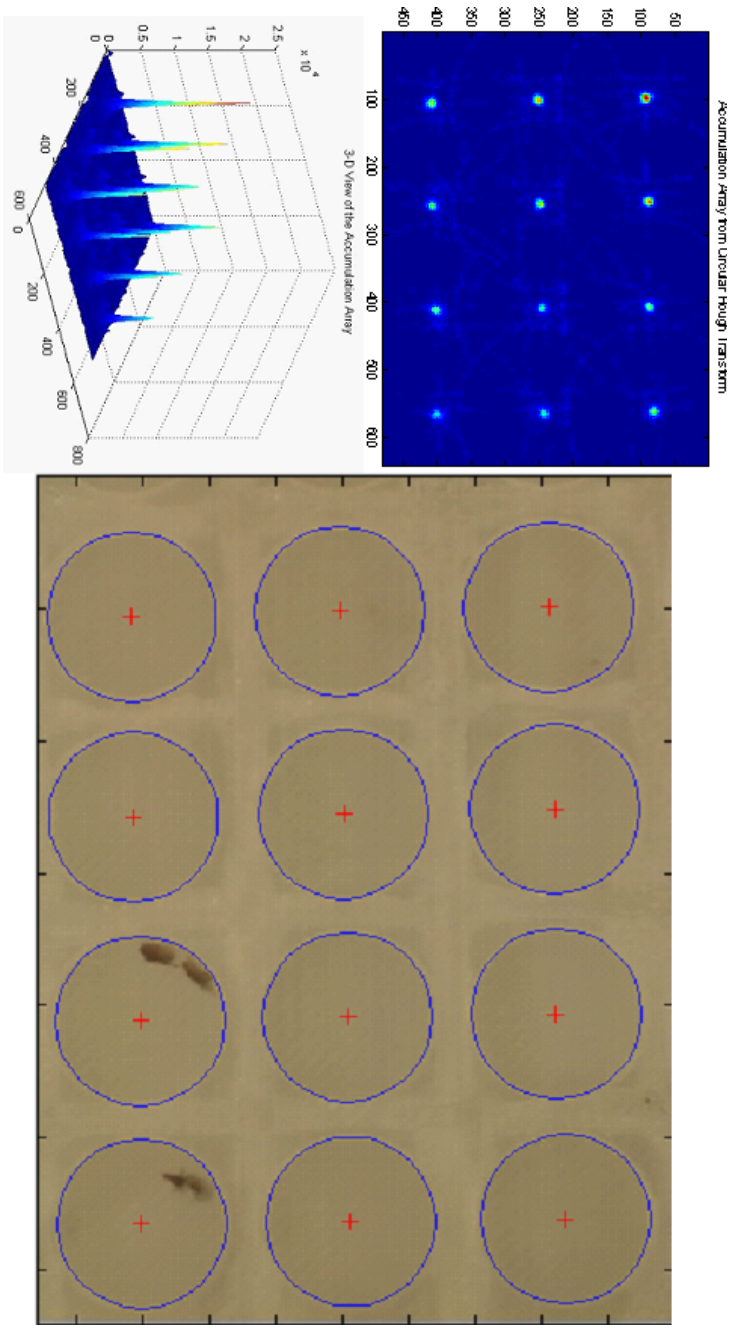


Figure 10: (left): accumulated votes in hough space as a heat map.. Top picture shows hough space from top and visualizes hough peak positions, bottom picture further visualizes hough peak heights and relative heights. (right) original background picture with detected circles. Circle midpoints are marked with a red cross, circles are drawn in blue.

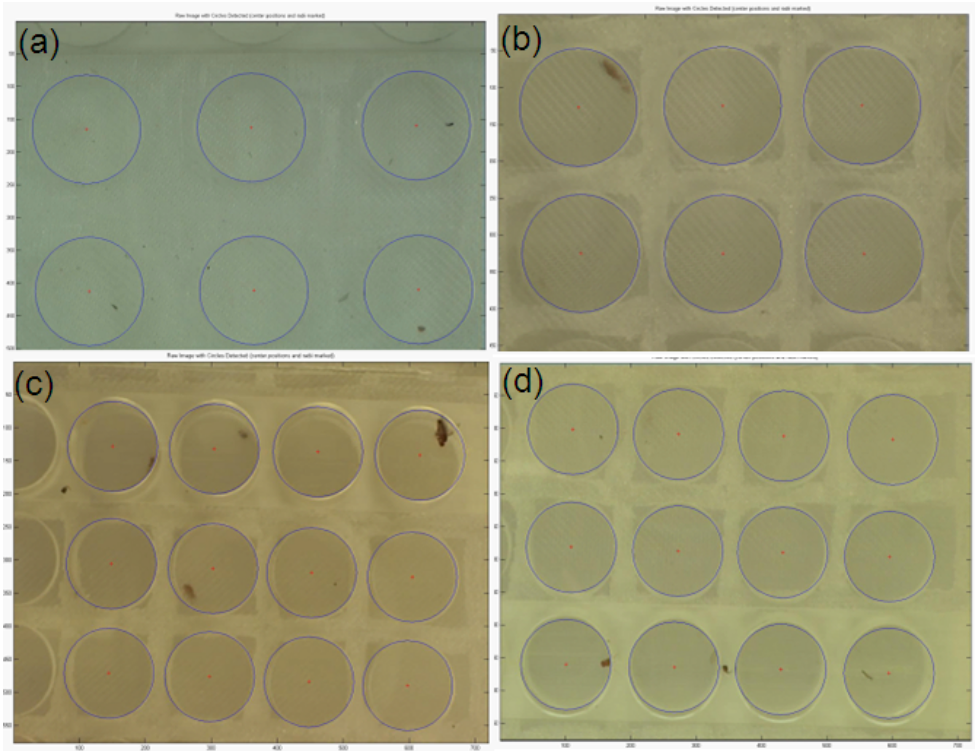


Figure 11: (a)-(d) Median backgrounds and detected arenas of four different recording setups

are too many circles for this discussion purposes, the algorithm was applied to a fragment of the upper left corner only (see blue lines) only, which shows just a few circles on first sight.

Figure 12 (b) and (c) show a gray level image of the upper left corner, detected circles are drawn in blue, circle centers are drawn in red. The right picture (c) shows *all* circle candidates. This includes

- strong evidence circles: the two fully visible concentric circles in the upper left region of the picture
- weak evidence circles: the algorithm treats the spiral-like shape as three pairs of concentric partially visible circle candidates and detects another partially covered circle in the middle
- poor evidence circles, the middle circle has two outer concentric circle candidates with very little evidence, the distance from its midpoint to the upper left circle, the distance to the shape in the right and the distance to an inner spiral circles are equal, furthermore the outer spiral border gives another candidate for the same midpoint[21].

Note that the upper left circle has two concentric candidates, inside and outside of the black ring. The shaded circle around that ring gives a hough peak that is less than ten percent of the highest hough peak, and is therefore treated as too poor to be detected. The circles is too blurred, it has no clear edge and therefore gives a wide but not too high peak in hough space.

The left picture (b) contains *filtered* circles only. In case of concentric circles the radius with the strongest peak is chosen, in case of overlapping circle candidates the candidate with weaker peak is removed. The picture therefore contains four circles, the circle candidate in the middle overlaps with other circles, and concentric circles are resolved by taking the strongest peak candidate.

Alternatively to the hough transform there are implemented arena detectors **by brightness** and **by movement**, the first one requires a clear brightness contrast between arena and non-arena areas of the background, the later one requires the flies to move and operates on overlayed movements, much like the one depicted in figure 15 in section 3.2.4. All implemented arena detectors may be combined to find agreed arenas only. However, since the brightness contrast may not always be given, and since mutant flies might move too little or not at all, the hough algorithm introduced above is the most reliable, most robust and default method. Other methods may optionally be selected in case videos fulfill additional quality requirements, however, but experience has shown that arena detection by the hough method is sufficient for standard videos.

The arena detection step is a major milestone during the **preprocessor** workflow (see section 3.1) it enables background smoothing, further quality

[21] The whole Kandinsky picture is full with such poor evidence circles, it is very likely that the exact spacing of objects has been done that way on purpose.

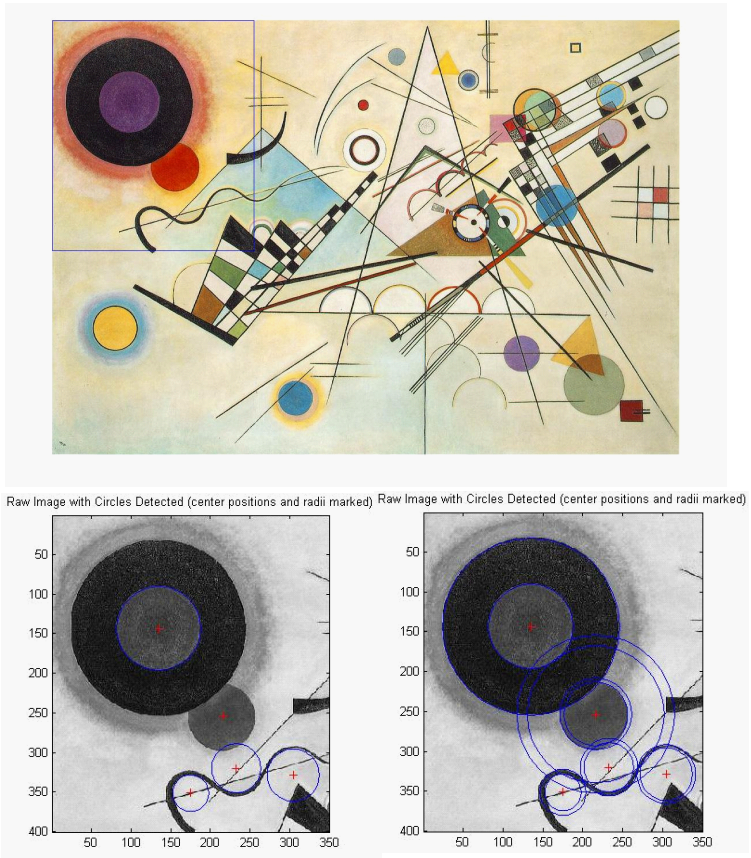


Figure 12: (a) Kandinsky Composition VIII, 1923. The upper left corner (blue rectangle) is used for further processing (b) filtered circle candidates in blue, midpoints in red. Overlapping and concentric circles are filtered and removed. (c) all circle candidates are drawn, some have weak evidence only.

control steps and to split the videos into pieces, each containing exactly one arena. During the **tracker** processings arena information is used to specify a region of interest, such that objects outside arenas do not influence the segmentation process. In the **postprocessor** module detected arena radii are used to normalize measurements, such that different camera zooms are compensated and videos are comparable to each other.

3.2.4 Quality Control Checks

Before invoking individual steps of the detection sequence several quality control checks are designed to automatically sort out videos that are not conform to the defined recording standards. This may be caused by disturbances during video recording, like e.g. movement of the camera or the arena plate, objects crossing the camera view or a wrong number of flies within a chamber.

Quality control steps are mainly performed within **preprocessor**, the idea is to reject unconform videos as early as possible. The illumination corrected set of evenly-spaced video frames D', which is also used for computation of background $bg_{D'}$, is sufficient for most major quality control steps. Although it contains just a set of snapshots of the video, it still allows to check arenas for the correct number of flies, possible movement of an arena or whether a video is distorted in an undesired way. Possible intrusions resp. arena infringement by objects from outside are checked in the preprocessor for all frames in $D\prime$ *and* later in the **tracker** for every single frame. This is to ensure that very short intrusions that are not visible in any of the frames in $D\prime$ are still captured, but also that longer and therefore more obvious intrusions are recognized early. The **postprocessor** performs a final quality control where it ensures that loaded tracking data meets all expected requirements. Although this is already guaranteed by current tracker implementation, the postprocessor still performs this quality control in order to be self-contained and to potentially allow postprocessing of otherwise tracked data.

3.2.4.1 Checking Flies per Chamber

The most common reason for video rejection are chambers containing the wrong numbers of flies. The system is defined to process chambers with exactly two flies. Some downstream processings[22] are specifically optimized for the two fly scenario, it is essential that the preprocessor rejects improper videos. However, since each chamber is individually downstream processed it is sufficient to reject the chambers with wrong numbers of flies only. This step therefore allows

[22] In particular, the foreground refinement and the wing region assignment of the tracker (sections 4.2.2 and 4.2.3), automatic resolving of occlusions and event detection of the postprocessor (sections 5.2.2 and 5.2.5) and the annotationTool are specifically optimized for the two fly scenario.

a partial rejection of videos, involving some undesired chambers only, while conform chambers may still be accepted for further processing.

The fly number control is applied on the set of distributed video frames $D\prime$, the number of flies in each of the frames is determined by segmenting the corresponding foreground image (as in the Tracker, see section 4.2.1.3). The size of all detected foreground regions is normalized by an estimated fly size, which is determined by the mean of all values between the 0.25 and 0.75 quantile of all region sizes where two regions were detected. This normalized value gives an estimation for the number of flies in the scene, discrepancies of up to 0.5 fly sizes are tolerated, more discrepancies lead to rejection of the affected chamber.

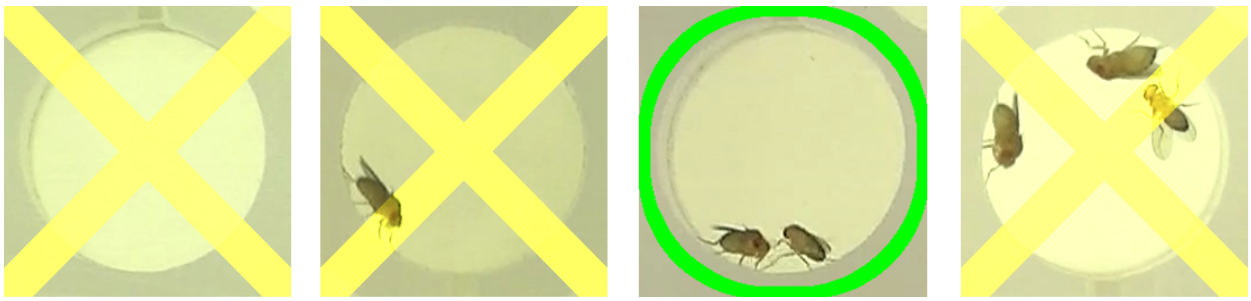


Figure 13: Snapshots of chambers with zero, one, two and three flies. The yellow cross marks rejected chambers.

Figure 13 shows quality control output for sample chambers with zero, one two and three flies, for chambers with wrong numbers of flies a rejection reason picture is generated, it shows the first invalid video frame in $D\prime$ as a witness and is marked as invalid with a yellow cross. Most invalid chambers have no flies at all, it sometimes happens on purpose that experiments are recorded with standard 11 or 14 chamber plates, although just a few chambers are filled with flies. Chambers with one or three flies are usually caused by human error, this happens in less than one percent of the cases. Incorrect chambers were rejected as expected for all videos processed so far, which involved more than ten thousand chambers.

3.2.4.2 Movement Detection and Boundary Intrusions

It is important to ensure that the videos are not moved, i.e. that both, camera and arena plate remain static for the whole video. Moved videos would result in ambiguous or blurred median backgrounds, which may irritate or even disable arena detection functionality, the first major milestone of the preprocessor.

Movement detection has to be called before arena detection (see preprocessor workflow section 3.1), the standard segmentation functionality as used for

tracking or for fly number control is not yet available at that early stage. Therefore the movement detection uses an alternative segmentation method (method *variance*, see section 4.2.1.1 p. 50) which detects foreground based on pixelwise mean and deviation values, $\frac{frame-mean}{deviation}$, and is thus good in detecting non-static frames.

The idea of the movement detector is to divide the set of frames in $D\prime$ into two parts, in particular the first and the last halves of the set are chosen. Then, for each half set D'_1 and D'_2 the pixelwise mean and deviation values M_1 and Σ_1 resp. M_2 and Σ_2 are computed, and deviation factors F are computed for each frame with mean and deviation *of the other half*, $\forall f_i \in D'_1 : F_i = \frac{f_i - M_2}{\Sigma_2}, \forall f_i \in D'_2 : F_i = \frac{fi - M_1}{\Sigma_1}$. In case of non-moved videos and therefore static backgrounds in all frames, the values M_1 and M_2 resp. Σ_1 and Σ_2 are expected to be similar and that therefore higher values in F denote non-static fly regions[23] only. However, in case the arena plate is moved w.l.o.g. in the *second half* of the video, the frames *after* that movement will have high deviation factors F within the otherwise static (low deviation factors) background. In other words, a moved arena plate causes parts of the background to appear as if they were foreground since they are out of normal (camera fluctuation) deviation range.

A video is rejected as moved, witnessed by a frame f_e, when the number of potential foreground pixels $|\mathfrak{F}_e|$, $\mathfrak{F}_i = F_i > foregroundThreshold$[24][25], differs from the mean of all $|\mathfrak{F}_i|$ by more than $rejectionThreshold$[26] standard deviations.

Figure 14 depicts the sample rejection image for a moved video, it shows the witness frame, which is the first frame identified to be inconsistent to the rest of the frames in $D\prime$. In case of detection of undesired movement *all* chambers of a video are rejected.

The principles of boundary intrusion detection are similar to movement detection although the task is quite different. For boundary intrusion control step checks arena infringement, i.e. it check whether objects from outside intrude the videos chambers. Such an intrusion would not only potentially occlude the flies or parts of an arena, it may interfere with the ongoing fly behaviour. Therefore arena borders are eagerly checked, besides the frames in $D\prime$ every single frame is checked for boundary intrusion within the tracker, and a detected intrusion immediately leads to a rejection of the affected chamber(s).

While some older videos, not recorded for automatic processing, sometimes contain fingers, hands or sheets with notes crossing the camera, most intruding objects are freely flying or crawling flies that escaped from previous experiments.

Figure 15 depicts the problem statement, it shows an overlay of fly movements that reveal that a fly crossed the scene and intruded several arenas.

[23] It is not necessary segment exact fly body regions, just the rough amount of overall foreground detected is sufficient for movement detection purposes.

[24] The parameter $foregroundThreshold$ is by default set to 1.

[25] $\mathfrak{F}_i$ is defined as a binarized mask.denoting potential foreground pixels.

[26] The parameter $rejectionThreshold$ is by default set to 2.

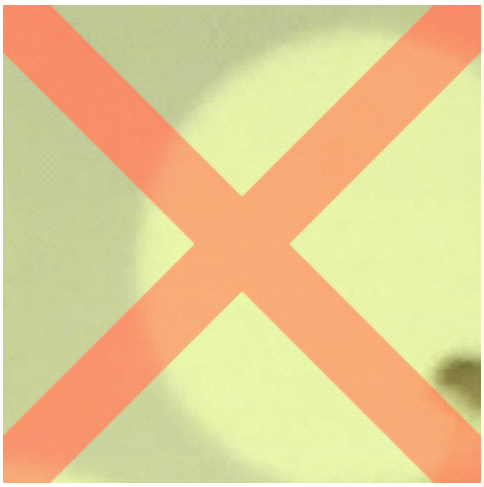


Figure 14: Witness frame for a moved video, the chamber is rejected and therefore marked with a red cross.

Boundary Intrusion detection also bases on the *variance* segmentation method (section 4.2.1.1 p. 50), and again the expected static background is checked for undesired potential foreground regions. This time a boundary, a protecting ring layed around the arenas borders, is eagerly watched in order to report intruding objects. This watched boundaries inner radius is half an average fly length $l/2$ longer than the arena radius, its outer radius is a another $l/2$ longer, the protecting ring has thickness of half a fly length and does not tangent its own arena in order to avoid wrong alerts.

Figure 16 depicts the automatically determined watched boundary zones of a sample video. It is assumed that outside objects have to cross the boundary zone before they disturb the arena. Using the same method and definitions as in for movement detection above, a chamber is rejected due to boundary intrusion in case its watched boundary contains a connected potential foreground region $\mathfrak{F}_i$ that is bigger than half an average fly size.

Figure 17 depict samples where intruding flies were detected within the watched boundary, affected chambers were rejected.and thus not further processed. The first picture shows a potential interaction that might influence the examined flies behavior. The last picture shows a flying fly, barely visible but still captured due to the boundaries sensitivity.

Boundary intrusion was observed in about one percent of all examined chambers.

Figure 15: Overlay of all fly movements within a video. Obviously a fly crossed crossed the scene, and interfered with chambers 3, 7 and 12.

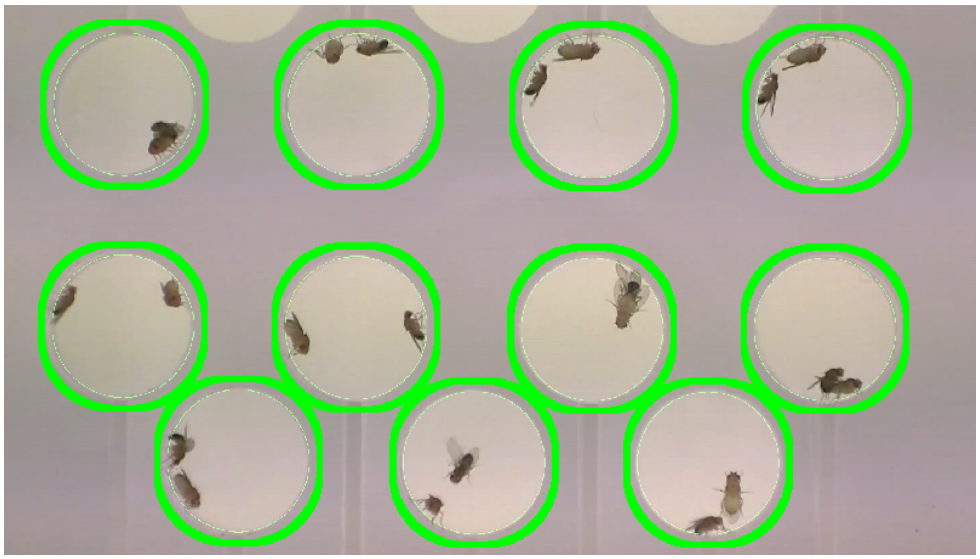


Figure 16: Snapshot of a video with detected arenas and watched boundary rings (in green).

Figure 17: Samples for detected boundary intrusions, rejection images contain the witness frame marked with a red cross.

3.2.4.3 Other

Quality control is further capable to detect distorted videos, i.e. videos that are not equally scaled in x and y direction. Distorted videos are highly undesired because recorded chambers are **ellipses** instead of circles, which may irritate circular arena detection, and flies have different dimensions dependent on their orientation. Such videos may be automatically rescaled and therefore corrected. Distortion of videos may be avoided by configuring camera and recording program settings correctly, auto-rescaling is turned off by default and throws a warning instead of auto-correcting the videos.

Figure 18 (a) depicts an automatically rescaled background picture with detected circular arenas.while (b) contains an original, unrescaled and thus distorted video frame. The arenas in (b) are distorted as well and have been drawn as ellipses with the detected distortion factor. When looking at a video frame like in (b) it is sometimes hard to see whether the picture is distorted or not, however, when comparing (a) and (b) it is very obvious that the arenas in (a) are circles and the arenas in (b) are not.

Besides distortion, incorrect camera settings or use of old camera models may cause image artefacts. Most common effects are either black or colored stripes at the video border. Figure 19 shows how such artefacts may disturb downstream processing, in particular (a) shows how sharp black border artefacts, which have high gradient values where they tangent the recorded picture, irritate the gray-scale gradient value based hough transform. The artefacts in hough space appear much worse than in the original picture, all hough peak values lie on an undesired line effect. Figure 19 (b) shows how blurred cyan border artefacts influence the background smoothing operation in an undesired way, the arenas on the right contain noticeable cyan color.

Since such border effects were quite common with certain old camera models, such effects are detected and affected videos are automatically cropped by the

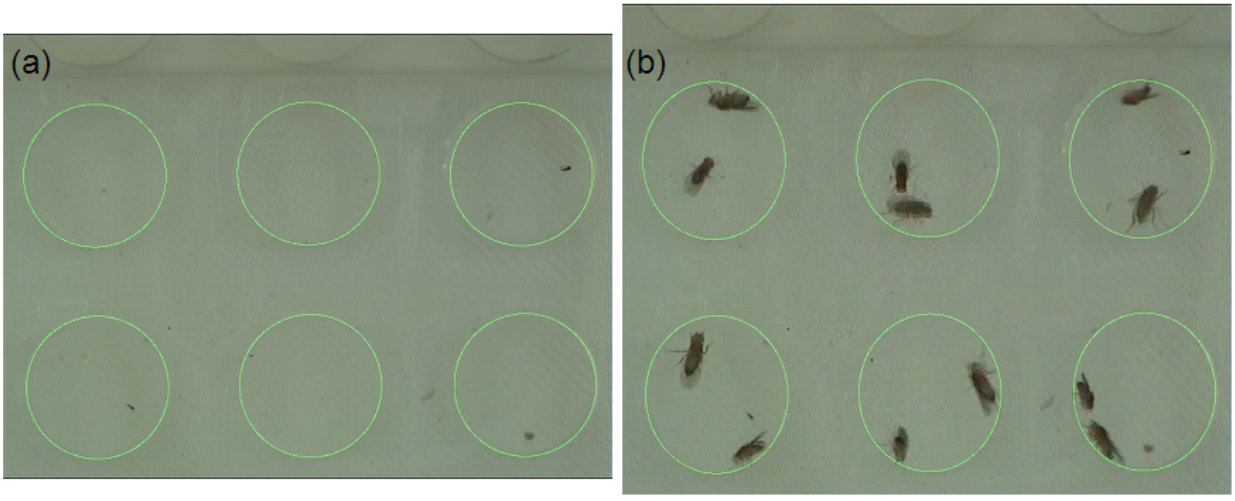


Figure 18: (a) rescaled background with detected circular arenas. (b) original distorted video frame with elliptic arenas.

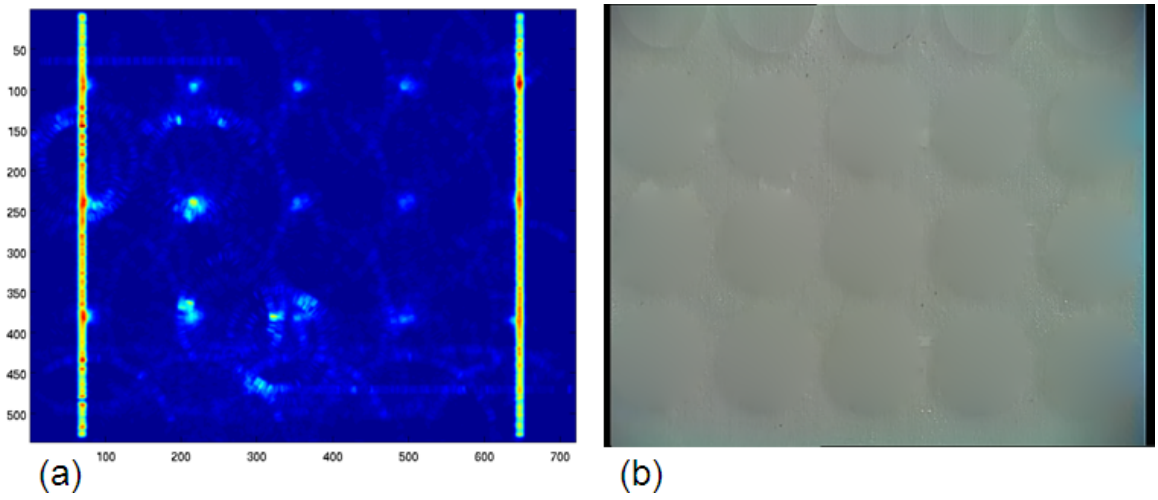


Figure 19: (a) Hough space of a circular hough transform, applied to a background containing sharp black border artefacts. (b) Smoothed background disturbed by blurred cyan border artefacts.

preprocessor. This reduces the dimensions of the recorded frames, however, since only artefact information is cut, no recorded video information is lost.

Finally figure 20 depicts the limits of arena detection. Figure 20 (a) shows that arenas may be placed at arbitrary positions, e.g. in a hexagonal instead of an rectangular pattern. All arenas are detected as expected. Figure 20 (b) shows that arenas touching the video border - even when they tangent it just a little - are marked as invalid. The corresponding circles are well detected, but arenas are removed since flies might walk outside the recorded scene. Further, such arenas would not be properly protected by a watched boundary. The detected arenas in (b) are considered to be correct.

The background in figure 20 (c) contains a loading bar, a tool that biologists use to fill flies into the arenas. Since the circles within the original background are poorly visible, and in contrast, the circles within the loading bar are very clear, the loading bar irritates the circular arena detection, which filters hough peaks that are less than ten percent of the highest detected peak. The clear loading bar circles therefore result in undetected normal arenas. Lading bars (or other circular objects within the background) are therefore highly undesirable, although arenas may still be detected correctly (as in figure 11 (d) in section 3.2.3).

A summary of all required recording standard is provided in the conclusion chapter, section 7 p. 176

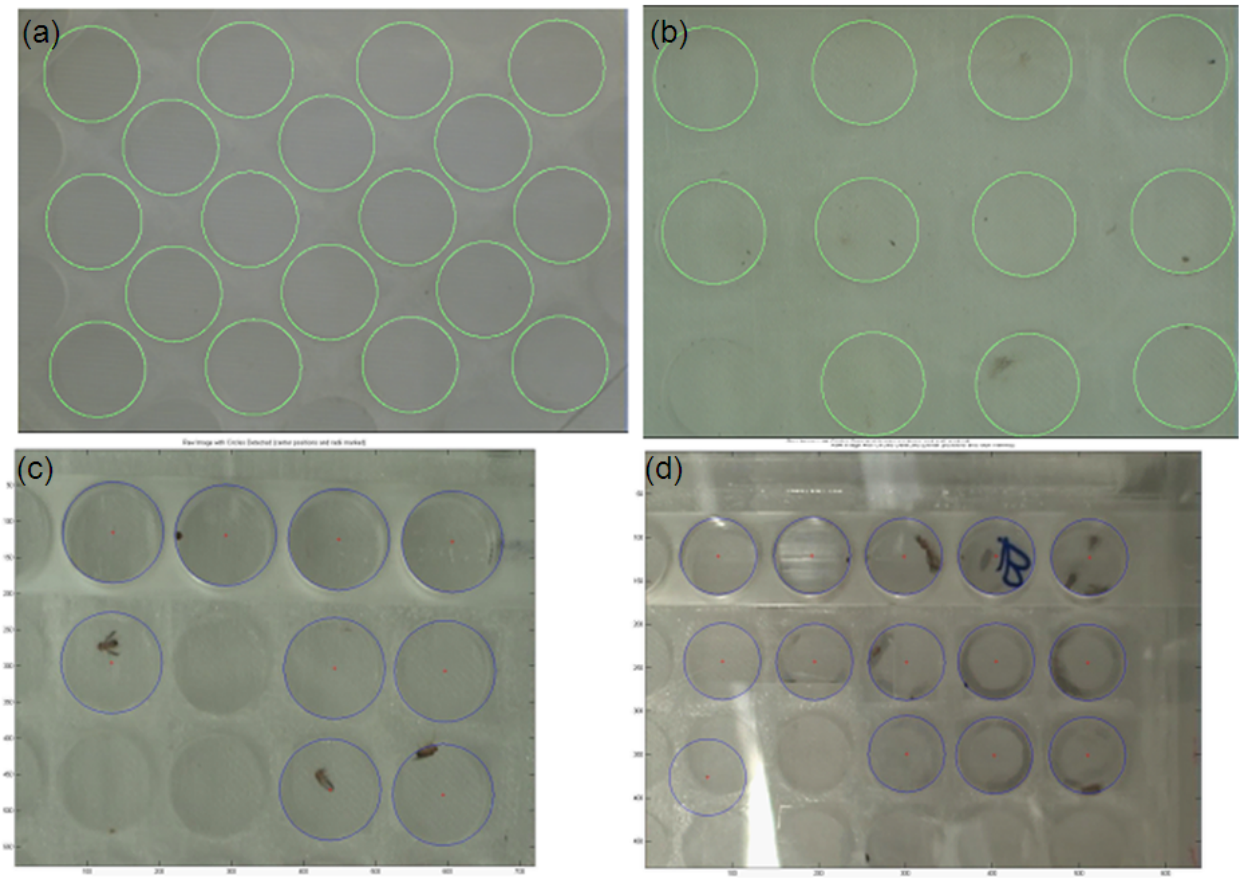


Figure 20: Arena detection samples, green circles depict correct and expected results, blue circles mark samples with undesired affects. (a) Background with non-rectangular arena placement. (b) Arena overlapping with video border. (c) Forgotten loading bar that affects arena detection. (d) Light reflection that affects arena detection.

4 Tracker

The tracker processes single arena videos generated by the preprocessor. It derives a foreground video by subtracting the smoothed and illumination-corrected background image from each frame. It uses these foreground images to segment and track fly bodies and fly regions, and to derive primary attributes for each fly in each frame.

The tracker component is the core element of the whole system. It captures the essence of complex video information in a much simpler time series representation. While the preprocessor component is mainly a supporting element, the tracker component contains image processing steps that reduce video frame images to a few relevant numbers, the *primary attributes* (see below, for more details c.f. 4.2.4).

The tracker module is also the performance critical part of the system as it processes huge amounts of video data. Tracking a video means to apply a sequence of image processing steps to every single frame, such that a video with 25 frames per second is translated into a time series with 25 entries per video second. Video frames are processed in order of occurrence, each time series entry is derived from the pixel values of its by order corresponding video frame.

The derived time series representation is a simplified but much more handy information representation. Even for simple computing tasks, like identification of the bigger of two flies, this comparison is much easier when operating with scalar area attributes rather than pixels in gray level images. Advanced analysis or extraction of biologically relevant attributes are therefore computed from time series data rather than raw video data, this reduces processing complexity and time.

The sections of this chapter are a survey through the image processing steps required to convert video data into time series data, section 4.1 guides through the workflow of image processing steps and 4.2 describes the involved methods.

It is the main purpose of the tracker module to extract this primary fly attribute information (see section 4.2.4), which enables all further downstream computations in the postprocessor (in chapter 5).

4.1 Workflow

The first and most important step is the segmentation method. It operates on background subtracted videos where static parts of a video frame are eliminated, such background subtracted video frame images contain foreground information only. Within these foreground images the foreground regions are segmented, the segmentation method combines a thresholding with a gradient correction approach (see section 4.2.1), it derives small and dark body and circumfluent wing regions. All following image processing steps operate with these regions in order to boost their quality or to guarantee their feasibility.

The segmentation section 4.2.1 introduces all required operations and definitions for body and wing region extraction, due to its constructive character the workflow of the default segmentation method *gradient*2 is described at the end of section 4.2.1, in figure 28 on p. 63. This figure and workflow descriptions particularly include definitions and terms introduced in 4.2.1.1 and 4.2.1.3, it describes the workflow how an input video frame is transformed into body and wing regions B and W, the preprocessor given background models bg_{SA} and bg_{sF} are incorporated in that process, which mainly consists of thresholding steps (see 4.2.1.1) and a gradient correction step (see 4.2.1.3), for more details c.f. figure 28 itself.

The workflow after the segmentation step is straightforward. The segmented foreground regions are refined (section 4.2.2) by incorporation of a priori knowledge about the number of flies per arena[27]. In case of multiple detected body areas within a single wing area an intelligent merge of these body areas may improve the quality of detected foreground regions. Refined body and wing regions are then assigned to fly objects (see 4.2.3). In case both fly bodies were detected within just one circumfluent wing area, splitting this wing area and assigning its separated parts to the fly bodies again improves detected fly objects.

Finally, main characteristics of these fly regions are then extracted (see 4.2.4) as primary attributes.

The resulting time series consists of a set of such primary attributes per frame and is a set of scalar numbers that were derived directly from each video frame image. Primary attributes found the information basis for all further downstream computation, all further processing results are directly or transitively dependent from these primary attributes.

4.2 Methods

4.2.1 Segmentation

"In computer vision, segmentation refers to the process of partitioning a digital image into multiple segments (...). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze." [13]

For each foreground frame two regions are derived, one segmenting the body region B of flies only, and one for entire fly which is referred as the wing region L, although it also contains the body region and often includes legs. From the wing region L an improved wing region W without legs is derived, legs are removed by a morphological opening operation. The morphological opening also removes small wing areas and optionally wing areas that do not contain body

[27] In general the number of flies per arena is required to be constant. Particularly for courtship videos the number of flies per arenas is further known to be exactly two, arena not conform to this requirement will be sorted out early by preprocessor quality control.

areas. Further holes are filled within the region and for ease of later operations it is ensured that wing regions W fully contain the body regions B, such that $W \cap B = B$. Formal definitions of *regions* and *areas* are provided later, in sections 4.2.2, 4.2.3 and appendix 7, intuitively an area is a set of connected points, a region may contain multiple areas.

The tracker software package contains implementations of different segmentation variants [28] with different properties, below in section 4.2.1.1 methods *direct*, *variance* and method $fast2$ are mentioned, section 4.2.1.3 contains a detailed description of the method *gradient*2, which is the standard method in use. Due to the modular design of the software package segmentation methods may easily be added, alternative methods are required to derive binary regions B, L and W.

4.2.1.1 Thresholding

All segmentation methods introduced in this section base on thresholding in gray level images. Thresholding is a simple but fast segmentation method where gray level images are translated into binary images, such that individual pixels are marked as *foreground pixels* if their intensity is above a given threshold value, and as *background pixel* otherwise. Obviously, a key part for this method is the computation of the threshold values themselves, the binary foreground-background assignments are straightforward consequences from these values.

Segmentation method *direct* operates on original video frames and aims to segment fly regions directly out of given video frames. Figure 21 (a) depicts an original video frame converted into a gray level image while (b) shows the value histogram of the gray level image in (a). A simple threshold may be found by Otsu's method [14] "which chooses the threshold to minimize the intraclass variance of the black and white pixels." [6]. The threshold in (b) is shown in red and separates the gray values by a threshold of value 181 out of 255 possible brightness values. Figure 21 (c) shows the corresponding binary images, where gray values greater than the threshold have a value of 1 resp. white, while smaller gray values are depicted as 0 resp. black. Note that the threshold in figure 21 mainly separates the bright inside-arena background from the darker outside-arena background since these values are quite similar and numerous. However, the white foreground region contains the flies, and by setting regions outside the detected arena to 0, essentially the flies remain.

[28] Actually even more than these four approaches are implemented, further alternative segmentation methods (see implementation of function *doSegmentation* on page 181) may optionally be used instead of the standard methods. This may enable processing of videos that are not conform to the required quality standards resp. have further restrictions or needs (c.f. e.g. section 6.3.2 "spin-offs" on page 169).

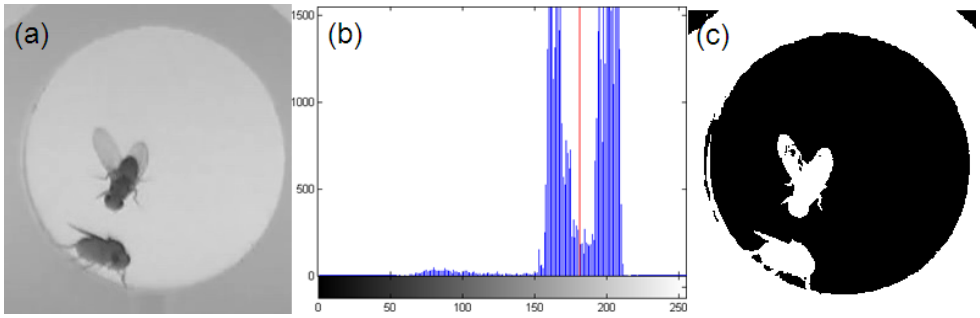


Figure 21: Direct Thresholding. (a) gray value image of original frame, gray values are given as values between 0 and 255. (b) gray value histogram of the image in (a), the height of a bar corresponds to the number of pixels with a particular gray value. The Otsu-threshold is marked in red. (c) binarized foreground region, segmented with method *direct*.

The fly region derivable from (c) corresponds to the region L (see p. 4.2.1), from which region W is straightforward computable. One approach to further extract region B out of L is to determine another *direct* threshold which Ortsu-separates *the gray values within L only*. However, while region L may be well approximated with method *direct*, segmentation of region B especially the border regions and included legs when flies are on the side shows its limits.

This method is introduced to demonstrate the basic principles of thresholding. It may provide an approximative overview about a scene, but is not suited for sharp and detailed segmentation. Method *direct* may for example be used to estimate the average number of pixels per fly early within the preprocessor, when more advanced background models are not available yet.

Alternatively to this - in method *variance* - the mean and deviation background model may be used to compute the deviation factor per pixel as $\frac{frame-mean}{deviation}$, such that a foreground image represents a pixelwise likelihood for being a foreground pixel. Again, thresholding may derive a binary foreground region.

Since RGB (Red Green Blue) colorspace is not linear in respect to color perception, the HSV (Hue Saturation Value) colorspace is more suitable for such statistical segmentation. While RGB colorspaces contain a red, green and blue value for each pixel, HSV colorspace contains its hue, saturation and value; fortunately images may be converted between colorspaces. Figure 22 visualizes the HSV color space as a color cone.

Segmentation in HSV colorspace videos is further more robust to shadow effects, as such effects disturb only the value component, while it leaves hue and saturation widely undisturbed. However, image operations within HSV colorspace are not as straightforward, for example a simple difference of images has to be computed differently for each component. While the value component is still computed by pixelwise subtraction of value components, the saturation

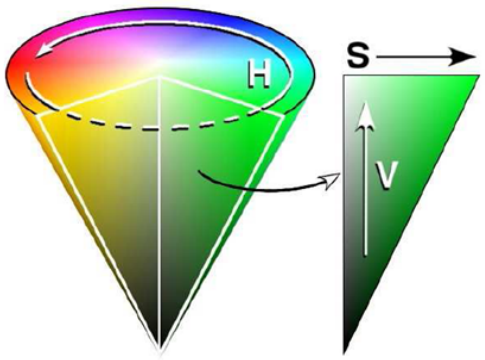


Figure 22: HSV color space as a color cone. Each color consists of a hue component H, a saturation component S and an intensity component V, source: [15].

component is computed as the absolute value of saturation differences, and the hue component, as it contains angle values, contains angle differences rather than simple subtractions. Simple statistical calculations, particularly for the hue component, quickly lead to circular statistics and Van Mises distributions, therefore processing HSV image is timewise much more expensive than dealing with RGB images.

For this reason statistical segmentation is used for quality control purposes in the preprocessor only (see movement detection and boundary intrusion in section 3.2.4), where its robustness against shadow effects is very desirable, while the performance critical tracker module operates with the much faster background subtraction and thresholding approach.

Segmentation method $fast2$ is computed significantly faster than the HSV based segmentation method above. Further, a major improvement comes with *background subtraction* of images rather than operating with original video images in method *direct*. Figure 23 (b) shows such a background subtracted image, it is derived by subtracting the background (c.f. figure 6 (b) in section 3.2.1, p. 25) from original frame in (a) and contains foreground information - essentially the flies - while static background information - essentially everything else - has been eliminated by the subtraction step. The main purpose of the preprocessor module is to support this segmentation step, it does so by providing appropriate background models for this subtraction step.

Thresholding of background subtracted images provides a more fine-grained segmentation, especially at the borders of the foreground region, which tend to be "shaky" when segmenting raw video images. The threshold itself is again determined from the gray level histogram, figure 23 (c) shows the original histogram computed from the background subtracted gray level image and its

shape-capturing moving average curve[29]. Note that this histogram and the derived curve capture the gray level distribution of the potential foreground regions, the background values are all very small and represented on the very left of the curve only. More details about characteristic points and thresholds within these histograms are provided below.

Figure 23 (d) depicts the binarized body regions segmented from these background subtracted regions, the detected regions are straight consequences of the determined threshold (see p. 52, the region according to the more conservative threshold is depicted here). In particular body regions are captured more precise than with method *direct*, especially when the flies are on the side. Note the missegmentation in chamber 11 at the lower right corner where a fly sits on the chamber ceiling glass with its stomach region towards the camera, such that the two darker head and tail regions are detected as bodies, but not the brighter stomacher area. This particular case may successfully be rescued by gradient correction (see section 4.2.1.3 below), otherwise an intelligent region merge algorithm (see section 4.2.2) would take care of this.

In general gray level histograms and especially smoothed histograms come in a *characteristic shape* for fly regions. Since figure 23 (c) is computed from the whole frame instead of a single arena only and therefore depicts the value distribution of *all* chambers, it captures this shape particularly well. Histograms may come in slight variations (see also figure 25) but do typically have stable characteristics.

When deriving a threshold for the body region the knowledge about expected value distributions is exploited and characteristic points in that value distributions are detected.

In particular, the slopes of gray level values are searched for the smallest negative slope within a window bordered by the largest negative slope plus a tolerance value of ten percent of the total space. The search is performed in both, the gray level histogram and the moving average filtered gray level histogram which contains the searched characteristic much clearer, and the smaller threshold is taken. In case of heavily distorted histograms and non-expected value distributions, which may occur for few single frames, the characteristics point may not be found and the threshold from the last frame is taken over, which makes this method both adaptive and robust.

Formally, the body threshold[30] is derived as follows

Definition 5 *Let O be a gray value histogram. A smoothed gray value histogram S is defined as a moving average filter with window length w, $S = movingAverage(O, w)$*

The window length w is by default set to ten percent of the histogram color space, which is rounded 25.

[29] The moving average is computed with a with window length of 25 gray value bars.

[30] The definitions below describe the body threshold for the default segmentation method gradient2, further introduced in section 4.2.1.3.

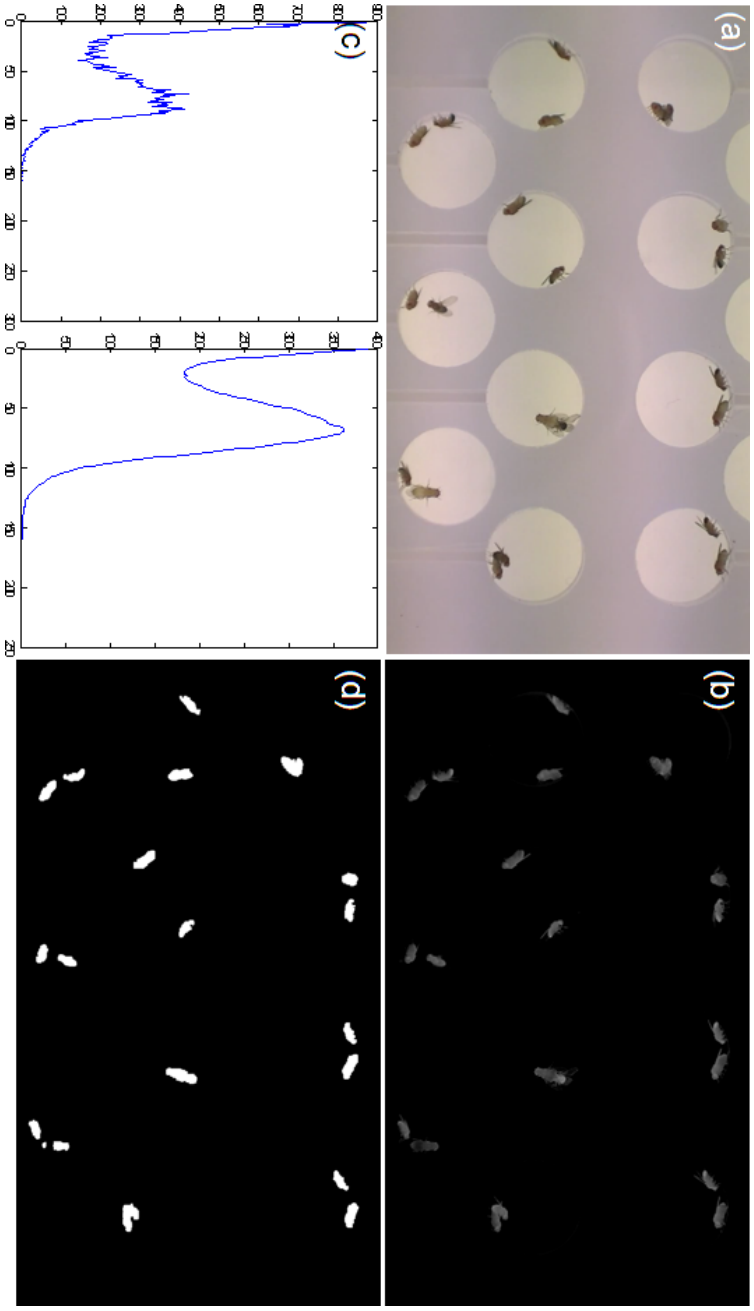


Figure 23: Thresholding Method. (a) original video frame. (b) foreground of video frame in (a), computed by background subtracting the smoothed background model bg_{SA}. (c) left: gray level histograms of the foreground image in (b), right: gray level histogram smoothed by a moving average filter with window length 25. (d) thresholded, binarized foreground region B, which contains the fly bodies.

Definition 6 *Let H be a histogram and ΔH be the gradients of that histogram. Further, let M be the indices for highest gradient values, $m = \{i|\Delta H_i = \max(\Delta H), i > w\}$ and m be the first index in M. The threshold candidate t_H is defined as $t_H = \{\{i|\Delta H_i = \min(\Delta H), w < i < m\}\}$*

Definition 7 *Let O and S again be an original and a smoothed gray level histogram. The body region threshold threshold t_{body} is defined as $min(t_O, t_S)$.*

For derivation of the wing threshold and wing regions[31] the detected body region and known relationships between the regions are incorporated. Figure 24 depicts the workflow of wing region segmentation.

Figure 24 (a) shows an original gray value image of a singing fly, (b) the background subtracted foreground image. Note that for wing segmentation the cautiously smoothed background bg_{sF} is used, further pixels outside the arena and pixels that were already detected as body region are set black. In (c) gray values are specifically adjusted such that the contrast between the background and the foreground is enhanced, this step is the key step for wing threshold detection. In particular the contrast is enhanced by mapping a small region of gray values to the full scale of values, this particular color region is dependent on the detected body regions. It is assumed that the brightest pixels are the ones directly neighboring the body region[32]. The number of these pixels are approximated by the number of pixels in a once dilated body perimeter. The intuition behind this step is, that the number of to be detected wing pixels may be estimated from the number of already detected body pixels, or more specifically, that the perimeter of body pixels is used to estimate the number of the brightest-to-expect wing pixels and to adjust resp. transform image intensity values accordingly. Formally the bounds for the contrast enhancement are defined as follows:

Definition 8 *Let B be the body region, P the perimeter of the body region, $d(P)$ the dilated body perimeter and $|d(P)|$ the number of dilated body perimeter pixels. Further, let I be the gray level foreground image and $|I|$ be the total number of pixels in I. The lower bound of the contrast enhanced color region is set to 0, the higher bound such that $\frac{|d(P)|}{|I|}$ of all pixels in I are saturated with the highest color value.*

This contrast adjustment may be implemented with a few lines of code in Matlab

Algorithm 9 *F = bgsF - I, F(~arenaMap | B) = 0;*
dP = imdilate(bwperim(B),ones(3));

[31] The formal definitions for wing thresholds and wing regions for segmentation methods $fast2$ and $gradient2$ are the same.

[32] This is especially the case since the transition between the body region B and the background is slightly blurred and the body region is selected conservatively. However, even with sharp body regions the wing region is expected to be directly connected to the body region.

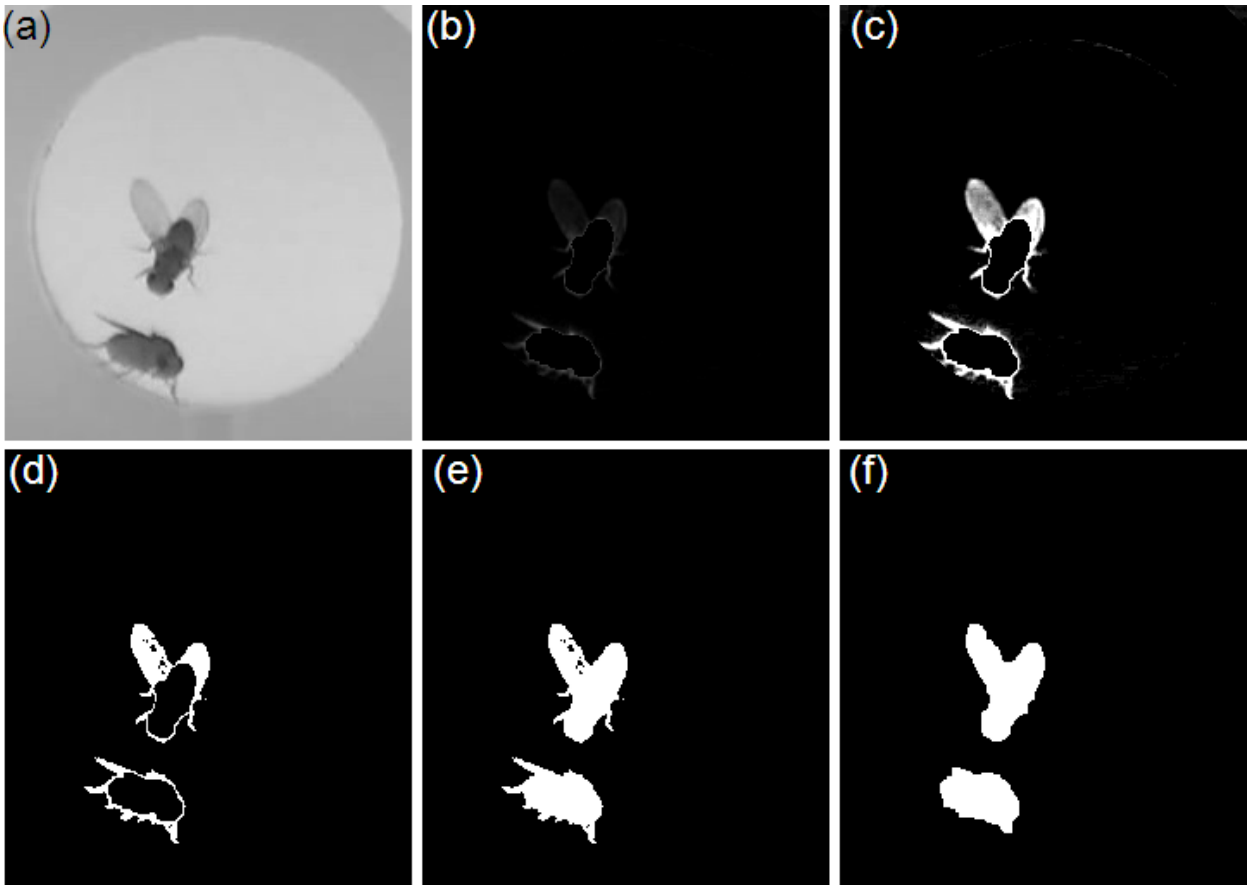


Figure 24: Segmentation of wing regions L and W. (a) original video frame as gray value image. (b) background subtracted foreground image. (c) intensity adjusted foreground image with enhanced contrast. (d) binarized wing region. (e) wing region L is the unification of (d) with body region B. (f) wing region W derived by morphological opening and closing of L.

Definition 10 ***Algorithm 11*** *wingPixelPerc = sum(dP) / length(dP(:));*
low_high = stretchlim(F, [0 1-wingPixelPerc]);
ret = imadjust(F, low_high, []);

The algorithm above describes how to derive the picture depicted in figure 24 (c). F denotes the foreground image, it is derived by subtracting the original video frame I from the cautiously smoothed background bg_{sF}, further values outside the arena or within detected body regions B are set to 0. Then dP is computed as the dilated perimeter of the body regions and $wingPixelPerc$ as the expected percentage of bright wing pixels. The command *stretchlim* computed the desired limits for the image intensity adjustment operation, in the example shown in figure 24 color values between 0 and 26 are adjusted to the full scale between 0 and 255. The last line computes the adjustment for the given limits.

Having a intensity adjusted image as in 24 (c), derivation of a segmentation threshold is straightforward. Otsu's method (introduced above) again delivers useful thresholds and experience has shown that - after the adjusting step - even a trivial threshold at the half of the colorspace gives suitable results.

Figure 24 (d) depicts a binarized foreground region, computed with a trivial threshold in the middle of the adjusted color space, it shows the wing region without the body region. Unifying this with the already given body region B, results in wing region with legs L, figure 24 (e), after filling holes and a morphological opening and closing operation wing region W - the wing region used for further downstream processing - is derived, see figure 24 (f).

Figure 25 shows some sample thresholds and segmentation results. In 25 (a) a sequence of pictures and the selected thresholds are shown. The samples in (a) show little histogram variation and a robust threshold selection. The samples in 25 (b) were chosen to show a variety of different pictures, different histogram shapes and therefore differently selected thresholds. Figure 25 (c) shows the very same pics as in directly above in (a), but thresholds are derived by method $fast2$ instead of $gradient2$. While the standard segmentation method $gradient2$ relies on two components, a seeding and a filling step, a rather conservative body threshold is sufficient. Since method $fast2$ does not perform any further correction of the body region the body thresholds aim to capture the whole body, and thus are - since wing thresholds are computed in the very same way - closer to the wing threshold. Therefore the visible gap between the two thresholds is noticeably smaller in (c) than for the more conservative method in (a).

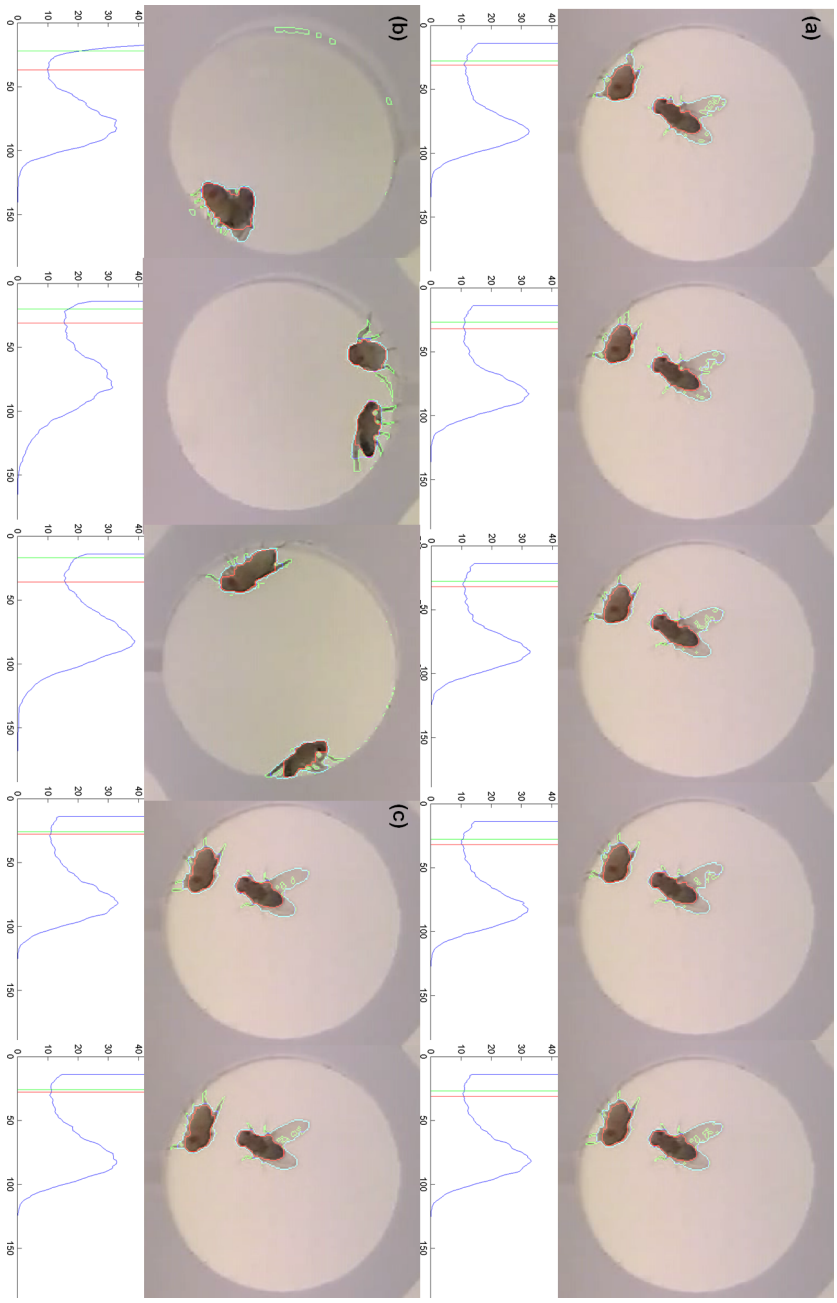


Figure 25: Segmented frames containing the original video frame, body region B (red), wing region W (blue) and L (green). The smoothed gray value histograms are depicted below, with body threshold (red) and wing threshold (green). (a) sequence of successive video frames, having similar histogram shapes. (b) alternate frames, showing different histogram shapes. (c) same frames as in (a) directly above, but method $fast2$ instead of $gradient2$ is used to determine thresholds.

4.2.1.2 Thresholding Discussion

Although thresholding is a widely used for segmentation in videos, naive thresholding comes with a number of known disadvantages.

A simple approach is to compute or accept a fixed threshold value[33] and apply it through the entire video. Such a timewise non-adaptive approach may be vulnerable to changes in background lighting, shades and (timewise) uneven illumination. The used thresholding method therefore computes **a threshold for every single video frame**, resulting in a more time consuming but adaptive thresholding variant. The series of threshold values further be stored, such that sequences of volatile threshold values are detectable and resolvable; outlier values resp. difficult cases may be replaced by neighboring results or recomputed by a different threshold determination algorithm.

Further, a naive approach is to apply a fixed threshold per video frame image, resulting in a regionwise non-adaptive method, again vulnerable to shades and (spatial) uneven illumination. The used thresholding method attacks this issue from both sides. On one side, **illumination and color correction** (see preprocessor, section 3.2.2) compensates regional illumination disturbances, which allows to keep threshold determination simple by using a single threshold value. On the other side, individual chambers of a video are detected and split within the preprocessor, such that the tracker module operates on single chamber videos only, which implicitly enforces computation of **chamberwise local thresholds**.

Thresholding is further known to be non-robust w.r.t. different video classes, i.e. a thresholding method providing a precise segmentation for a given video class may have different properties on fundamental different input videos. Although the used method was tested with a wide number of camera settings, it is not guaranteed to be applicable for arbitrarily recorded videos. Different video classes may require different segmentation methods; in addition to the methods introduced in this section dealing with standard videos (e.g. depicted in figure 27), alternate methods for different recording setups (e.g. depicted in figure **??**[34]) were implemented and are configurable.

Finally, naive thresholding may be error-prone when dealing with similar gray level values in background and foreground e.g. when dealing with dirt regions, and it may have difficulties to provide sharp segmentation for regions with blurry edges.

However, thresholding is simple and fast - and the approach is to exploit these features in the tracker module while compensating its disadvantages by one-time computations in the preprocessor mainly in the background model generation process.

[33] When evaluating different trackers e.g. the commercial software from Noldus had an interactive support for definition of such a *constant* threshold, then applied it to entire videos.

[34] The underlying video came in lower resolution, having blurred images, different lighting conditions and no contrast between chamber and non-chamber regions.

4.2.1.3 Thresholding Combined With Gradient Correction

In order to overcome all known disadvantages, the thresholding segmentation method may be combined with an alternate segmentation approach. Below a combined method named *gradient*2.

A complementary method is to segment according to the *edges* of an image, edges are regions with high gradient magnitude (see figure 26). The key idea is to first find a *conservative* threshold, with is easy to compute and underestimates the size of segmented foreground regions, and then to extend these smaller regions up to their edges. Regions surrounded by edges are filled out as foreground as long as they have a *seed*, a marker for regions of interest, coming from the previous thresholding step. Combining these two methods is timewise more expensive[35], but comes with benefits in segmentation quality.

The thresholding step deriving the body region for method *gradient*2 (see section 4.2.1.1 p. 52 above) is implemented as a conservative approach that typically underestimates the full extent of the body and wing areas. To fill out these regions, the segmented areas are used as seeds for a morphological reconstruction within the inverted gradient image, this operation fills out regions up to their bordering edges, much like a flood fill operation.

Figure 26 (a) and (b) show the gradient values X and Y in x resp. y direction, these pixelwise gradient information founds the information basis for edge detection. The gradient magnitudes G in (c) are derived directly from the pictures in (a) and (b) and are defined as follows:

Definition 12 *Let X and Y contain the pixelwise gradient values in x and y direction and x_i and y_i the gradient values for a particular pixel i. The gradient magnitude values G are derived by the pixelwise operation $g_i = \sqrt{x_i^2 + y_i^2}$, $G = \bigcup_i g_i$.for every pixel i.*

Figure 26 (d) visualizes contrast enhanced gradient magnitudes. High values indicate edge candidates, the bright borders of the fly regions are used as limits for a filling operation. The gray values in (d) are further binarized, using the median gradient magnitude of the body region as an upper bound, $E = G < median_{b \in B}(g_b)$, formally:

Definition 13 *Let G be gradient magnitudes, G_B be the gradient magnitude values within fly body areas B and $\tilde{X}$ denote the median of all values in X. The complement of the binarized edges are defined as $E = G < \tilde{G_B}$.*

The filling step itself is carried out by a morphological reconstruction operation. "Essentially a generalization of flood-filling, morphological reconstruction processes one image, called the marker, based on the characteristics of another

[35] Method *gradient*2 has by its definition more than double the costs of method $fast2$, its current implementation is about 3-4 times slower than $fast2$.

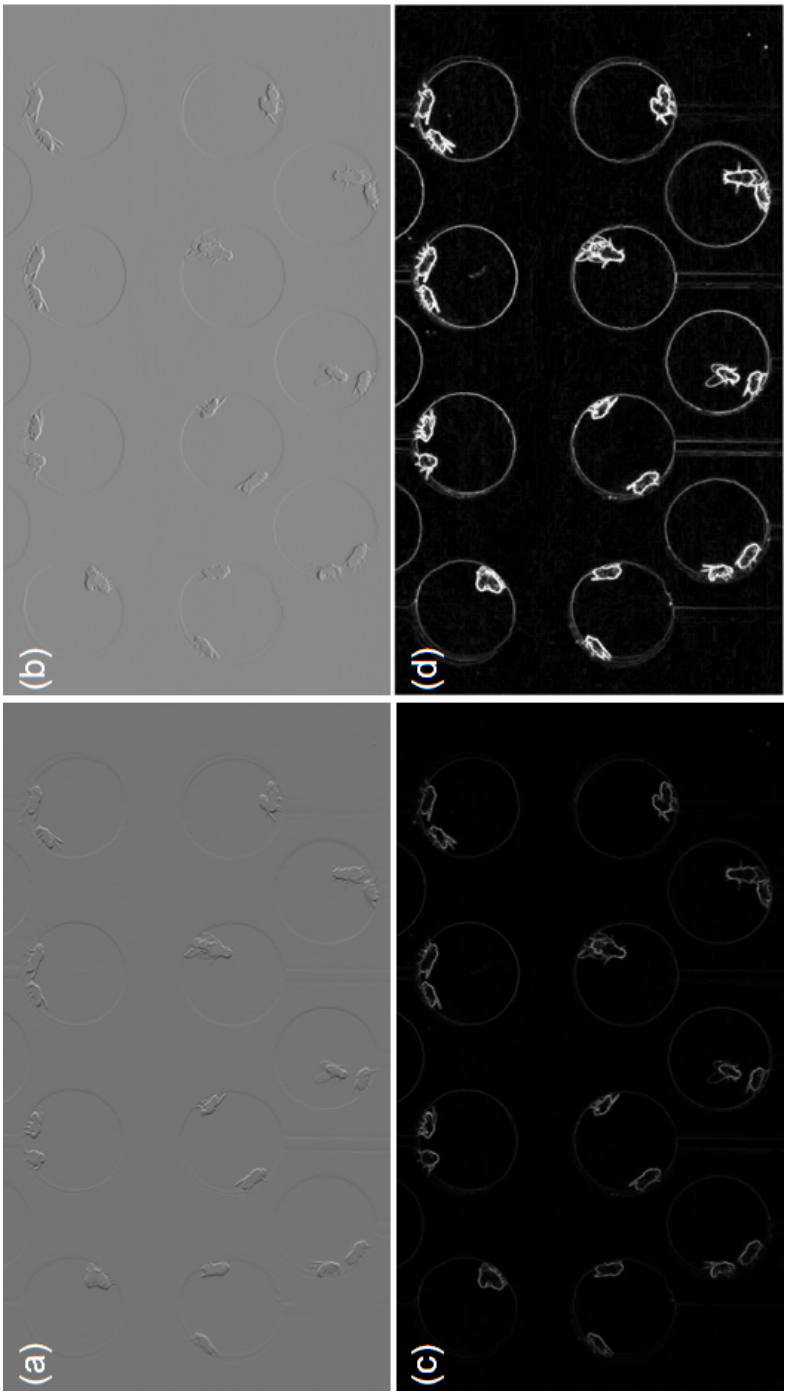


Figure 26: Gradients and edges of fly regions. (a), (b) gradient images X and Y in x resp. y direction. (c) gradient magnitudes G depicting the edges. (d) contrast enhanced gradient magnitudes, bright regions mark borders of fly regions .

image, called the mask. The high points, or peaks, in the marker image specify where processing begins. The peaks spread out, or dilate, while being forced to fit within the mask image. The spreading processing continues until the image values stop changing." [16]. For gradient correction purposes the marker, which contains the seeds of the filling operation, is set to $marker = B \cap E$, while the mask is set to the binarized edges complement $mask = E$, the morphological reconstruction using $marker$ and $mask$ carries out the desired correction.

Figure 27 shows two sample frames and their segmentation results with method *gradient*2. The introduced method is despite the typical problems of the thresholding approach robust to encountered every-day recording variations. The two videos in figure 27 were recorded with different camera setting, one using the highest and one the lowest possible brightness setting. The segmentation results in (a) and (b) indicate that especially the body region B (red) and the wing region W (blue) are reliably detected, the wing region with legs L (green) is depicted just to demonstrate that legs may be captured as well, but is not currently used for downstream processing. Note that the sample video frame in (a) is the very same video frame that was used for figure 23 on p. 23 and that the missegmentation in 23 (d) chamber 11 is corrected by the gradient correction step, see the red body region in 27 (a) chamber 11.

Figure 28 describes the workflow of segmentation method *gradient*2 and summarizes the major steps introduced in sections 4.2.1.1 and :4.2.1.3.

Figure 28 (a) depicts a **single chamber video frame** converted into gray values. In order to derive the background subtracted image for body segmentation in (b), the rigorously smoothed background model bg_{SA} (c.f. p. 3.2.1, figure 6 (b) and figure 28 (l)) is subtracted. The aim of this step is to eliminate background information, but to still treat the *whole* arena content, including static patterns that provide enough contrast, as a potential foreground region, such that a "lossless" candidate body region is segmented.

The **background subtracted gray level image** in (b) is translated into a gray level histogram and a moving average smoothed histogram curve and the thresholds for segmentation of body region B resp. wing region L are derived. Figure 28 (c) depicts the **gray level histogram** in black, the smoothed curve in blue and the **two thresholds** in green and red. The red threshold is used to derive the seeds for the binary body region B in (d)[36], the green threshold is derived from wing foreground in (g) but here visualized in the histogram of the body foreground (b), it is used to straightforward-derive the binary wing region L depicted in (h). Both thresholds are derived according to the definitions in 4.2.1.1, p. 52f.

In this particular example the gradient correction step contributes only one more pixel to the seeds picture derived by the red threshold in (c); 28 (d)

[36] Since the seeds for the body region B and the body region B itself are essentially identical - there is only one pixel difference in this particular sample - the body region is depicted only once and both, the seeds and the final region refer to subfigure (d).

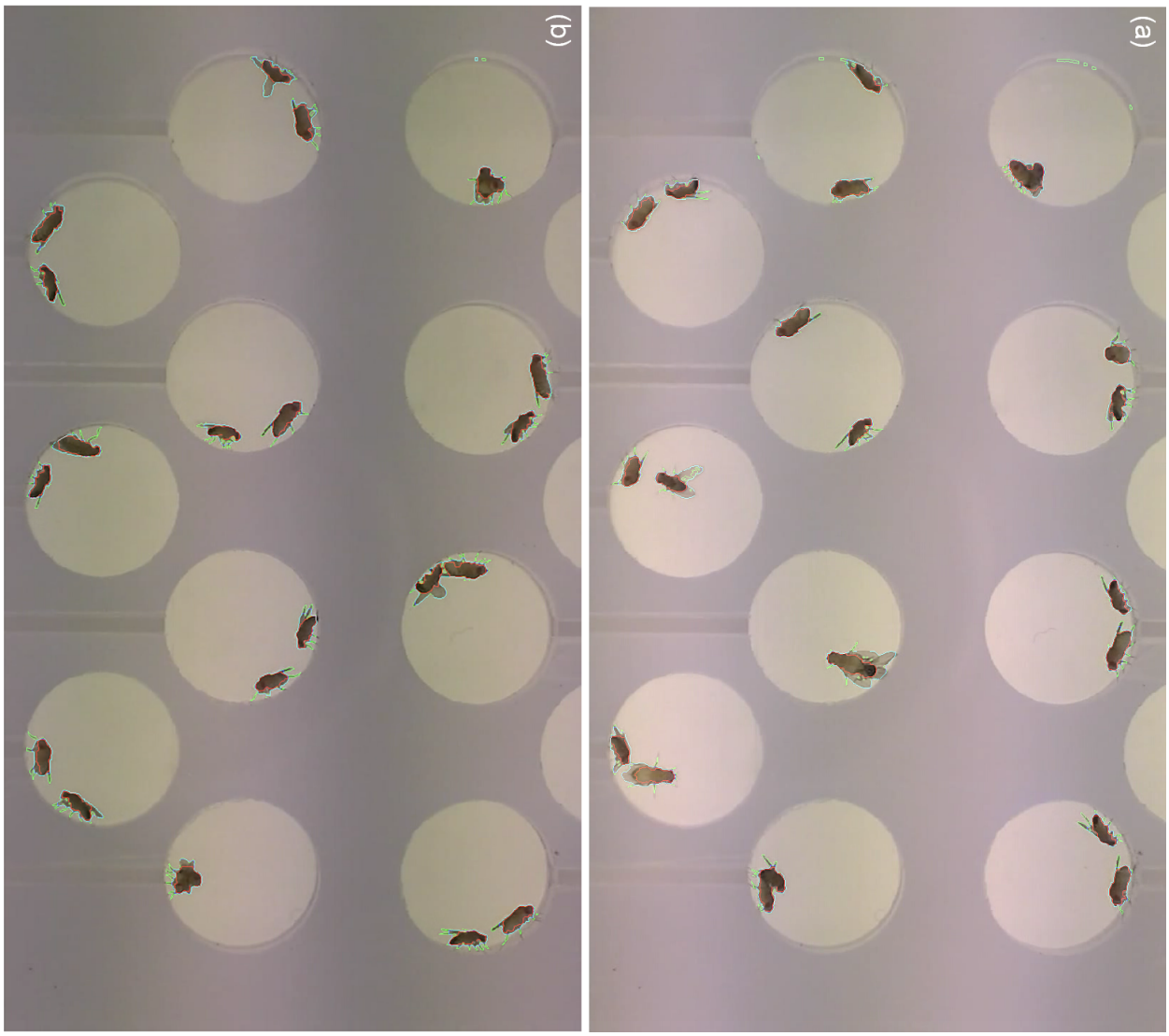


Figure 27: Segmentation results for video samples recorded with different brightness settings. (a) Camera setting for brightness set to plus 2 (maximum value). (b) Camera setting for brightness set to minus 2 (minimum value).

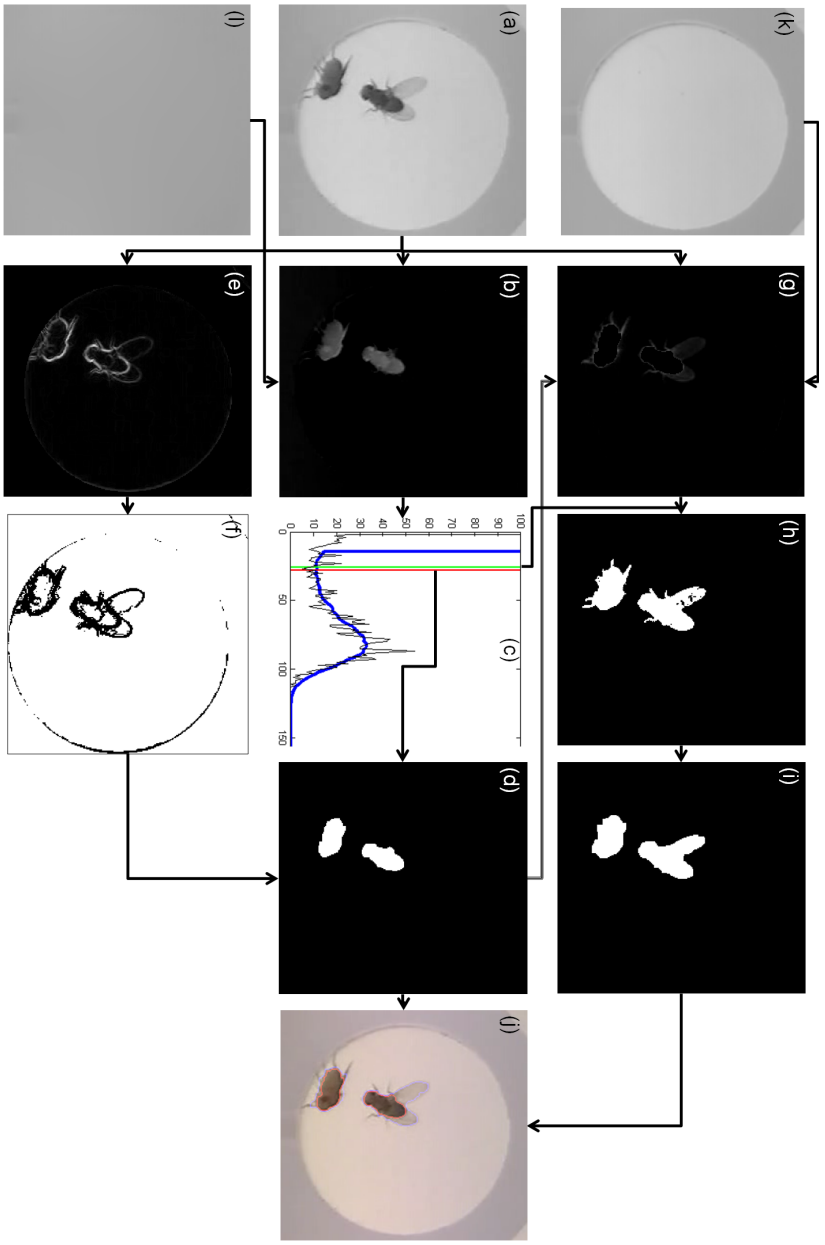


Figure 28: Workflow of segmentation method *gradient2*, black arrows indicate direct dependencies. (a) gray values of an original frame of a single chamber video. (b) background bg_{SA} is subtracted for body region segmentation. (c) symbolizes the threshold determination step, it depicts the gray value histogram (black line) the smoothed histogram (blue line) the body region threshold (red line) and the wing region threshold (green line). (d) binarized fly body region B. (e) gradient magnitude values, bright values indicate edges. (f) *mask* region for morphological reconstruction. (g) background bg_{sF} is subtracted for wing region segmentation, body and non-arena regions are black. (h) binarized wing region with legs L. (i) fly wing region W without legs and with filled holes. (j) original image with segmentation results, body region in red, wing region in blue. (k) cautiously smoothed background bg_{sF}. (l) rigorously smoothed background bg_{SA}.

therefore essentially depicts the seeds for gradient correction *and* the whole body regions B.

However, the gradient correction step in general is a major improvement and visualized in (e) and (f), (e) depicts the gradient magnitudes, edges are characterized with bright values, and (f) depicts the complement of the binarized edge region E that is used as *mask* for the morphological reconstruction operation. The morphologically reconstructed and thus **gradient corrected body region** B in (d) is one of the two result regions.

Figure 28 (g)-(i) depict the steps for wing region segmentation as described in section 4.2.1.1, in particular in figure 24. Figure (g) shows another background subtracted foreground image, for wing segmentation the cautiously smoothed background model bg_{sF} (k) is subtracted from the original video frame (a), further the body region B and pixels outside the arena are set to black. The (green) wing threshold from (c), derived from (g) as described on p. 4.2.1.1[37], is used to derive the wing region with legs L in (h), a morphological opening and closing operation removes legs and fills holes in order to derive the **wing region without legs** W in (i), the second of the two result regions.

In figure 28 (j), the original video frame is overlayed by the segmented body region B in red and the segmented wing region W in blue, these two regions compose the segmentation result that is used for further downstream analysis.

The input background models bg_{sF} (k) and bg_{SA} (l) are given as by the preprocessor and used to segment a video frame (a) into result body and wing regions (d),(i),(j). When segmenting with the rigorously smoothed background model bg_{SA} the aim is to detected every potential foreground information within an arena, even static flies or dirt pieces are detected by this step. This part of the segmentation is particularly specialized to detect the body region, wings having same color as surrounding background may be lost here. When segmenting with the cautiously smoothed background model bg_{sF} the contrast of flies to the arena background is exploited, the background model captures fine-grained background information and within the background subtracted foreground image the wing areas clearly visible by eye. Body areas are eliminated[38] such that threshold search is not distracted by bright body pixels. This part of the segmentation is particularly specialized for wing region detection; body pixels are later added to complete the region.

Method *gradient*2 is the system's default segmentation method. It delivers foreground regions B and W - and if desired even L - and their underlying thresholding values. In rare cases where histograms are heavily distorted and threshold points are not found the threshold value of the last frame is used, in very rare cases where no body seeds are detected the less conservative method $fast2$ is used instead; both of these difficult and rare cases are detected and resolved automatically. In the test scenarios where segmentation quality of regions B and W was visually evaluated, method *gradient*2 turned out to be the best compromise between **runtime** costs, the tracker is the time critical com-

[37] and used in figure 24 (d)

[38] Body region pixels and also pixels outside the detected arena are set to black, gray value 0.

ponent, segmentation **quality** and a pragmatic **robustness** to given variations in recording setups.

4.2.2 Foreground Refinement

The segmented foreground regions B and W provide good estimations about body and wing regions, but do not come with any guarantees about the number of connected areas detected. However, incorporation of a priori knowledge about the number of flies per arena[39] allows to further refine the segmented regions such that plausible foreground regions containing at most two connected body and wing areas are guaranteed after the refinement step. In particular the body region B may contain dirt pieces or multiple detected body areas within a single wing area, e.g. in case of a stomach-up recorded fly, like in chamber 11, figure 23 (b), p. 23. This section describes an intelligent merge operation for multiple body areas within a single wing area that may further improve the quality of detected foreground regions.

At first the wing areas are refined. If more than two wing areas are detected this is caused by contaminated chambers. Wing areas that do not contain a body area are removed, and since the number of wing areas cannot exceed two, the smallest wing areas and their inner body regions are kept being eliminated in case that more than two wing areas remain. This refinement step guarantees that either one or two wing areas remain, the first case occurs in case the wing regions of the two flies overlap, the latter case otherwise.

The foreground refinement step is optional, but useful if it is assured that exactly two flies are given. It particularly improves body detection within wing areas, especially when flies are upside down (bright stomach). Here it may happen, that multiple, but mostly two parts of the body are detected within the wing regions, and it is desirable to merge these regions into one body region.

In order to correct such a fragmentation of body regions, segmented body regions may be connected with a convex merge operation, which fills gaps between two areas with values of their convex hull. If exactly two wing areas are detected, this procedure is applied to all body areas within each wing area. If a single wing area is detected, the smallest body region is iteratively merged with its closest region until exactly two body regions remain.

Formally the merge operation is defined as follows:

Condition 14 *Given that it is known that there are exactly two flies in a chamber.*

For binary images the simpler representation is used where pixels with color value 1 are enumerated within a set of points.

[39] In general the number of flies per arena is required to be constant. Particularly for courtship videos the number of flies per arenas is further known to be exactly two, arena not conform to this requirement will be sorted out early by preprocessor quality control.

Definition 15 *A region R is a set of two-dimensional points p, $R = \{p_i = (x_i, y_i)\}, |R| = i$. The empty region $\varnothing$ and set operations $\cup$, $\cap$, $\setminus$, $-$ are defined as usual.*

Definition 16 *Let R and S be regions and $\widehat{R}$ the convex hull of region R. Then the binary operation convex merge $\uplus(R, S)$, written as $R \uplus S$, is defined as $R \uplus S = (\widehat{R \cup S}) - ((\widehat{R} - R) \cup (\widehat{S} - S))$.*

Conclusion 17 *Commutativity of $\uplus$ directly follows from the commutativity of $\cup$.*

Lemma 18 *$\uplus$ has identity element $\varnothing$.*
Proof. $R \uplus \varnothing = (\widehat{R \cup \varnothing}) - ((\widehat{R} - R) \cup (\widehat{\varnothing} - \varnothing)) = \widehat{R} - (\widehat{R} - R) = R$ ■

Definition 19 *Let $\mathfrak{R}$ be a set of $n \geq 1$ regions, $\mathfrak{R} = \{R_1, ..., R_n\}$. The operation bulk convex merge $\uplus(\mathfrak{R})$, written $\biguplus \mathfrak{R}$, is defined as*

$$\biguplus \mathfrak{R} = R_1 \uplus \biguplus_{R_i \in \mathfrak{R} \setminus R_1} R_i$$

Conclusion 20 *It follows for every region R that $\biguplus R = R$ and in particular that $\biguplus \varnothing = \varnothing$.*

Condition 21 *Given that it is known that there are exactly two flies in a chamber and further given that at most two connected wing areas were detected.*

The second requirement comes for technical reasons, however it may easily be guaranteed by the wing region elimination step introduced at the beginning of the section.

It is distinguished between the following cases.

case a: two wing areas detected.

In this case two separate wing regions W_1 and W_2, $W_1 \cup W_2 = \mathfrak{W}$ were detected. The aim is to guarantee that there is exactly one body region per wing region.

Definition 22 *Let $\mathfrak{B}_1 \subseteq \mathfrak{B}$ be the subset of body regions in wing region W_1, $\mathfrak{B}_1 = \{B | B \subseteq W_1\}$, and $\mathfrak{B}_2$ the subset in wing region W_2.*

The merged body region is then defined as

$$\mathfrak{B}' = \biguplus \mathfrak{B}_1 \cup \biguplus \mathfrak{B}_2$$

case b: one wing area detected

In this case only one wing region was detected and it is known that both body regions are within that one wing region.

Definition 23 *Let $R_i \in \mathfrak{R}$ be a region, $a_i \in A$ the Area* [40] *of that region, $c_i \in C$ the Centroid of that region and $d_{ij} \in D = \{|\overrightarrow{c_i, c_j}|\}$ the distances from Centroid c_i to all other Centroids.*

Definition 24 *Further the smallest Area is defined as $a_s \in A = \{a_i | a_s \leq a_i\}$, the smallest region as $R_s \in \mathfrak{R}$, the shortest distance from Centroid c_s as $d_{st} \in D, d_{st} \leq d_{sj}, s \neq j$, the closest centroid to c_s as c_t and the region closest to the smallest region as $R_t \in \mathfrak{R}$.*

Given body regions $\mathfrak{B}$ containing $n = |\mathfrak{B}|$ regions with $n \geq 3$ [41]. Then s and t are determined according to the definitions above and the smallest body B_s and the body closest to the smallest body B_t, $B_s, B_t \in \mathfrak{B}$, are replaced by $B_s \uplus B_t$.

Definition 25 *Let $\mathfrak{B}^0 = B$. It is defined that $\mathfrak{B}^{i+1} = (\mathfrak{B}^i - (B_s^i \cup B_t^i)) \cup B_s^i \uplus B_{B_t}^i$*

Since $|\mathfrak{B}^{i+1}| = |\mathfrak{B}^i| - 1$ for $0 \leq i < n - 1$, defining the resulting merged regions as $\mathfrak{B}' = \mathfrak{B}^{n-2}$ guarantees exactly two body regions within the wing region.

[40] Areas as number of pixels

[41] The other cases are trivial: $|\mathfrak{B}| = 0$ may be excluded due to the given condition, since every founded wing region encloses a body region, $|\mathfrak{B}| = 1$ is a standard occlusion case with a single body and a single wing region shows how to deal with that. The case where $|\mathfrak{B}| = 2$ is trivial since two body regions were immediately found , therefore no further merge operation is to be done.

Algorithm 26 *merge smallest to closest*
$distMatrix =$
$\mathfrak{B}'_{hat} = \mathfrak{B}_{hat}$
$while|\mathfrak{B}'_{hat}| > 2$
$i = \mathfrak{B}_{ihat}|\min(\mathfrak{B}_{hat})\%withsmallestregion$
$j = \mathfrak{B}_{ihat}|\min(\mathfrak{B}_{hat})\%closestregionTosmallest\,\mathrm{Re}\,gion$
$\mathfrak{B}_{new} = \mathfrak{B}_i \cup_{\mathfrak{C}} \mathfrak{B}_j$
$\mathfrak{B}'_{hat} = (\mathfrak{B}'_{hat} - \{\mathfrak{B}_i, \mathfrak{B}_j\}) \cup \mathfrak{B}_{new}$
$end - while$

According to the two conditions above there are no further cases to be regarded, it is straightforward to guarantee that these conditions hold. The default implementation marks frames with more than two wing regions as invalid.

4.2.3 From Regions to Flies

In a further processing step refined body and wing regions are assigned to fly objects. In order to assign body regions and wing regions to each other, a *BoundingBox* is computed for every body and wing area, a *BoundingBox* contains the smallest rectangle that encloses the region.

Formally, a bounding box may be characterized by the upper left and the lower right point, always consist of vertical and horizontal edges.

Definition 27 *Let R be a binary region $R = \{p_i = (x_i, y_i)\}$. A bounding box $B_R = (tl, br)$ is defined by the two points $tl = (\min_{p_i \in R}(x_i), \min_{p_i \in R}(y_i))$*

and $br = (\max_{p_i \in R}(x_i), \max_{p_i \in R}(y_i))$.

Definition 28 *Let B_R and B_S be bounding boxes. Bounding boxes B_R is encloses bounding box B_S, denoted $B_S \subseteq B_R$ whenever $tl_R <= tl_S$ and $br_R >= tr_S$ for both x and y values.*

Definition 29 *Let R and S be binary regions and B_R and B_S the corresponding bounding boxes. Region R is defined to enclose region S, denoted $S \subseteq R$ whenever $B_S \subseteq B_R$.*

At first each body region is assigned to an in individual fly object. Then each wing region is then assigned to fly objects whichs body region it encloses, this enclosing relation is approximated by corresponding *BoundingBox*es, see definitions above.

In case every wing area is assigned to exactly at most one body area this fly object assignment is very straightforward. However, in case both fly bodies were detected within just one circumfluent wing area W, and therefore $B_1 \subseteq W$ *and* $B_2 \subseteq W$ partitioning this wing area W into two parts $W = W_1 \cap W_2$ and assigning its separated parts W_1 and W_2 to the fly objects with bodies B_1 and B_2 again improves the detected fly objects, it makes sense to split the wing area between the two fly body areas with a proper split operation.

For this reason the **competitive floodfill operation** is introduced, it is defined much like the regular flood fill operation, a special case of a morphological reconstruction, where a region is filled from a set of seed points up to its borders, given by a a binary mask. The intuition of a *competitive* floodfill operation is that two different types of seeds, e.g. in different colors, start filling a given region such that already colored points are removed from the mask and therefore not colored again. When applying this operation on a to-be-partitioned wing region W, $mask = W$, initialized with the two body regions B_1 and B_2, $marker_1 = B_1$ and $marker_2 = B_2$, the competitive floodfill operation assigns each pixel of the wing region to the geodesic closest body region.

Formally, this competitive floodfill operation is defined as follows.

Definition 30 *Let R be a binary region. The morphological dilation $d(R)$ is defined as a pointwise operation for each point in $p \in R$, such that each point in $d(R)$ is the maximum of all points in its 8-connected neighborhood.*

Definition 31 *Let R and M be binary regions and $d(R)$ be the morphological dilation operation. A geodesic dilation $g(R, M)$ w.r.t. mask M is defined as $g(R, M) = d(R) \cap M$.*

Definition 32 *Let R and M be binary regions and $g(R, M)$ be the geodesic dilation operation. Further let $g^0(R, M) = R$ and $g^{i+1}(R, M) = g(g^i(R, M), M)$. The morphological reconstruction $r(R, M)$ of mask M is defined as $r(R, M) = g^\intercal(R, M)$, where $g^\intercal$ denotes the least fix point with $g^\intercal(R, M) = g^{\intercal+1}(R, M)$.*

Definition 33 *Let R_1, R_2 and M be binary regions and $g(R, M)$ be the geodesic dilation operation. The competitive floodfill separation f is defined as $f^0((R_1, R_2), M) = (R_1, R_2)$ and $f^{i+1}((R_1, R_2), M) = f(f^i((R_1, R_2), M), M \cap C^i)$, where conflicting region C^i is the intersection of the two regions returned by f^i, $C^i = \bigcap f^i((R_1, R_2), M)$, and $f^\intercal$ is again the least fix point with $f^\intercal((R_1, R_2), M) = f^{\intercal+1}((R_1, R_2), M)$.*

The competitive floodfill operation used is computed using the two enclosed body region regions B_1 and B_2 as seeds for the filling of the circumfluent wing region W, such that $(W_1, W_2) = f^\intercal((B_1, B_2), W)$. In order to provide a full partitioning of W potential conflict points $\bigcup_{i=0}^{\intercal} C^i$ may be assigned to wing region W_1.

Figure 29 shows a sample wing region, depicted in (a) contains two body regions, depicted in (c). The partitioning suggested in (b) is derived by the flood fills predecessor method, a voronoi splitting operation, where every point of the wing region was assigned to the closest body region, for distance measures the euclidean distance instead of the geodesic distance in floodfill separation was used. Although voronoi separation performs well in most cases, the particular case depicted in 29 (b) reveals the weakness of the method, the wing tip of a fly is misassigned to the other fly since the direct line to that fly is shorter.

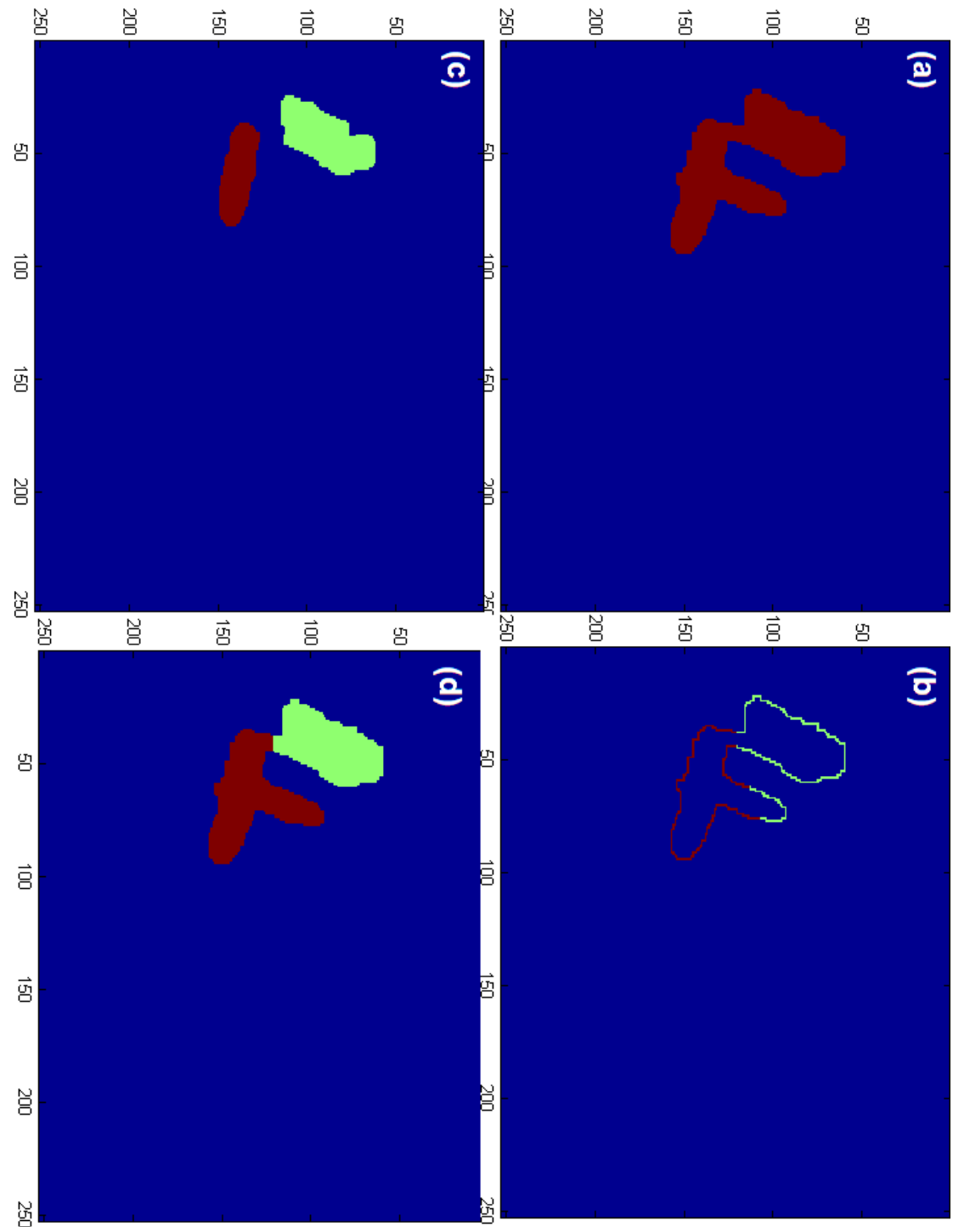


Figure 29: Approaches for wing region partitioning. (a) original wing region W. (b) wing region partitioned by voronoi-splitting operation. (c) body regions B_1 and B_2 used for competitive flood fill initialization. (d) floodfill-separated wing regions W_1 and W_2, colored in the corresponding body colors in (c).

The subfigure (d) directly below shows the superior result derived with the competitive floodfill separation, the floodfill-separated wing regions W_1 and W_2 are colored like their corresponding body seeds B_1 and B_2 in (c). The wing area in (a) is properly split„ the resulting regions associated with resulting fly objects are improved and depict useful information, in this particular case an otherwise not detectable singing fly.

This section introduced an approach that allows to successfully divide overlapping wing regions in case of separate body regions.

4.2.4 Primary Attribute Extraction

From the fly objects derived in section 4.2.3, a fly is defined to consist of a single body area B and a single wing area W, the time series is derived, the time series is the result of the tracking process and consists of *primary attributes* per video frame. These attributes derived **directly from the picture** and are used to derive all downstream attributes, no matter how advanced they are, and build the interface between the tracker and the postprocessor modules.

For each *non-occluded* frame, that is a frame where two body regions were detected the primary attributes are extracted as described below, in case of *occluded* frames, frames where only one merged body region was detected, primary attributes are interpolated from neighboring primary attributes as described in section 5.2.4.

The refinement steps in 4.2.2 and 4.2.3 ensure that every non-occluded frame consists of exactly two flies, each having exactly one body and one wing area, primary attributes for each of these four areas B_1, B_2, W_1 and W_2. In particular, ellipses e_i that have the same normalized second moments are fit for each body area B_i, further ellipses we_i are fit for each wing area W_i. The following primary attributes are thus defined for each fly in each non-occluded frame:

- *Centroid*: the center of gravity for an ellipse e.
- *Orientation*: the orientation angle of e (0 is horizontal)
- *Area*: the number of pixels of that region
- *BoundingBox*: the smallest rectangle enclosing the region
- $MajorAxisLength$, $MinorAxisLength$ and *Eccentricity*: of ellipse e, the eccentricity is defined $Eccentricity = \sqrt{1 - \left(\frac{MinorAxisLength}{MajorAxisLength}\right)^2}$ and gives values between 0 and 1, 0 characterizes a line while 1 characterizes a circle.
- *Perimeter*: all perimeter points, derived as the dilated region minus the region; $Perimeter(R) = d(R) - R$. Since the amount of perimeter points may vary the dimensions are chosen according to the longest perimeter, non-used points are filled with zeros.

The attributes listed above are extracted for *both,* body *and* wing region, for wing regions a further attribute *Solidity* is derived, which is the number of pixels in *Area* as percentage of the number of pixels of of the regions convex hull. Optionally, the system may derive a *legTouch* attribute, which is true in whenever both wing region W are connected in the wing region with legs L, however, this attribute is currently not used for downstream analysis.

These listed attributes, for body and wing regions, form the set of default primary attributes. The set contains redundant information for convenience reasons, Eccentricity for example may also be derived from MajorAxisLength and MinorAxisLength, the interpolation section 5.2.4 exploits this relationships. All of these primary attributes are derived, unless otherwise noted, by the Matlab's *regionprops* operation (c.f. regionprops in [6]).

Note that primary attributes for individual frames may be reported as *suspicious* (see p. 122), and the downstream postprocessor may decide to rather interpolate attributes for a particular frame than to use suspicious values.

5 Postprocessor

The postprocessor searches for biologically relevant patterns within the tracked time series information.

The postprocessor component is the "brain" of the whole system. It derives advanced attributes from the simple information basis containing primary attributes[42] only. While the tracker component supports the postprocessor by providing simple primary attributes and is therefore hiding the complexity of image processing, the postprocessor component derives more advanced attributes and searches for *meaningful patterns* within these attributes to derive a few biologically relevant numbers, the *events* (see section 5.2.5).

5.1 Workflow

As a first step the tracking data is loaded into an easy to process data structure and in case of multiple chambers separated and sorted by chamber. Since the postprocessor may in principle run independently from the tracker it makes sense to ensure that all assertions required for downstream processing hold. Required assertions are particularly a constant number of flies per chamber and analysis of one chamber at a time. The preprocessor and tracker components presented earlier already guarantee these properties (c.f. section 3 p. 15 and 4.2.2), however, in order to be self-contained the postprocessor additionally verifies the quality of the time series it operates with.

For this reason the postprocessor checks the number of flies per frame, and merges, eliminates or fills up objects if needed. The merge operation used is simpler as the one introduced in section 4.2.2, as it operates on primary attributes only and not on entire foreground images, it mainly sums the area and computes the combined center of gravity. The elimination operation first separates and removes static, then small objects. In case only one fly region is detected the data structure is filled up with *missing* values.

After the tracking data is loaded it is *normalized*, and secondary attributes which are directly derived from primary attributes are computed. Since the primary attributes are measured in pixels and are therefore video dependent, a normalization step is necessary to enable comparison of data from different videos. The arena detection step in section 3.2.3 delivers essential information for this. As a constant size of recorded chambers[43] is assumed, the diameters of the detected arenas [in pixels] may be used to convert all length scales from pixels into millimeters resp. square millimeters, the frame rate of the video[44] may be used to convert time scales to seconds.

[42] Primary attributes are numbers derived directly from a video frame image, they found the information basis for all further downstream computation.

[43] The size of a chamber [millimeters of diameter] may be configured once for a whole class of videos, the default chambers had a diameter of 10 mm.

[44] The default frame rate is 25 pictures per second, however, the frame rate may be read directly from the video in most modern video formats.

While some secondary attributes are straightforward computations, e.g. the distance between two flies may be computed from their coordinate position, other attributes like the distance moved by a fly since the last frame or the flies velocity require identification of flies through longer sequences, section 5.2.1 summarizes the fly identity assignment steps, in particular 5.2.1.2 describes how to re-identify flies in unoccluded sequences while section 5.2.1.3 and section 5.2.2 show how to handle occlusions.

The postprocessor deals with *occluded* and *non-occluded* sequences differently, section 5.2.1.1 further contains a formal definition for occluded and non-occluded σ resp. τ sequences.

Figure 30 depicts differences in the information flow between such σ and τ sequences. As mentioned above, the postprocessor loads the primary attributes coming as results from the tracking module, performs some quality checks and deals with missegmentations (see section 5.2.3.1). Having a valid time series, the frames are partitioned in to occluded and sequences (see section 5.2.1.1), identities are assigned for non-occluded σ sequences (see section 5.2.1.2) and secondary attributes, these are attributes that derived directly from primary attributes, are computed. Based on these attributes a size-based method for resolving occlusions is invoked (see section 5.2.2.1) and a s-value dependent on the probability that fly A [45] in a sequence is the smaller and thus the male fly (see section 5.2.2.2).

For τ sequences two *possible worlds* (see section 5.2.4), which are branches of the time series for different occlusion assignments, are created and the primary attributes are interpolated for both worlds resp. for both possible occlusion assignments (see section 5.2.4.1). More particular, the characteristics of an ellipse that covers a fly body region are interpolated across frames where the only one merged body region was detected by the tracking module. Several candidate trajectories for the ellipses, mainly differing in the rotation direction of the ellipse, are compared against the observed merged body region and the best fitting pair of ellipses resp. interpolations is chosen.

Secondary attributes and fly identities are computed, based on these interpolated primary attributes, for every world. For τ sequences t-methods are invoked for resolving occlusions (see section 5.2.2.1), the resulting t-values (see section 5.2.2.2) depend on the probability that fly A in the σ sequence before the examined τ sequence is also fly A in the σ sequence after the τ sequence

The resulting s-values and t-values are fed into the SDH-Algorithm (see algorithm 67 in section 5.2.2.2) which derives the most plausible identity assignment for the flies throughout the entire video.

Having this identity assignment, the branched possible worlds may be merged back into a single time-series by selecting the data of the world that resembles the occlusion assignment and discarding the data of the other world. After this step a single full set of attributes is available for *all*, occluded or non-occluded frames, and flies may be sorted by their identifiers in all frames.

[45] Fly A is arbitrarily chosen as the left fly in the first frame of a σ sequence.

Based on these attributes the head and the tail are identified for each fly (see section 5.2.3.3) and all advanced attributes, which found the basis for the event detection step, are computed[46]. Section 5.2.5 describes how a bottom-up classifier computes the probability for courtship behaviour in every frame and further introduces top-down classifiers for several courtship subbehaviours, namely following, orientation, circling, wing extension and copulation. Such detected events are visualized in color coded *ethograms* for every fly, these ethograms provide a summary of the biologically relevant information within a video.

Main purpose of the postprocessor is to quantify and visualize biologically relevant information out of the tracked time series.

[46] Note that there are up to 2000 different attributes are defined and usable for training purposes of bottom-up classifiers. However, having a trained classifier that selected a particular set of observables, it is sufficient to compute the incorporated observable attributes only.

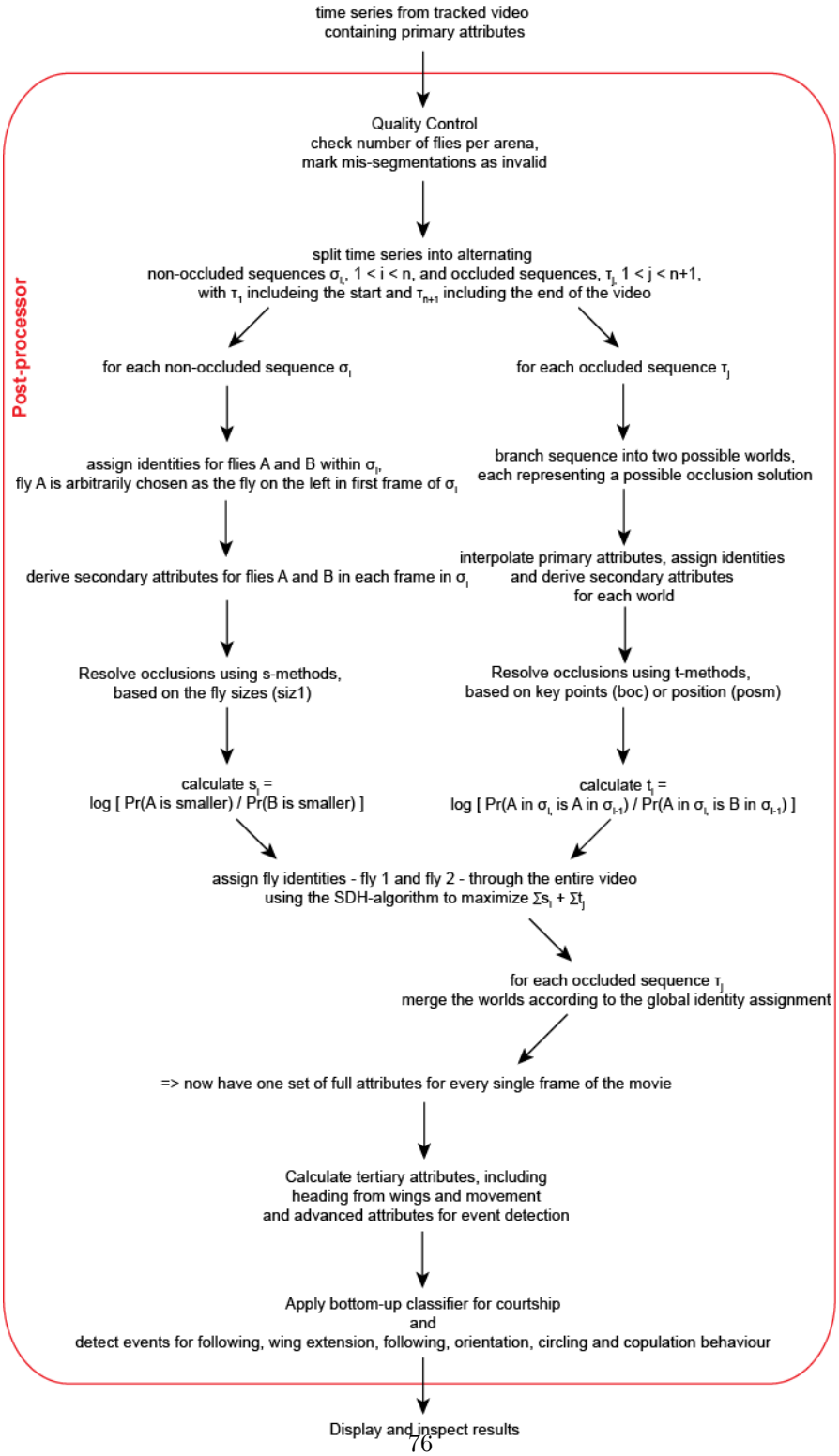


Figure 30: Main information flow through the postprocessor, with focus on handling of σ and τ sequences.

5.2 Methods

5.2.1 Identity assignment

The system aims to identify individual flies throughout the entire video for a variety of reasons.

Fly identification through consecutive frames is essential for computation of particular attributes, like e.g. the distance moved by a fly since the last frame or a flies velocity.

From a higher level perspective it allows to analyze and protocol courtship behaviour for each individual fly (see figure 63 p. 157) or to protocol spatial occurrence of characteristic events, even from the perspective of a particular fly (see 73 p. 172). It enables to design experiments where e.g. flies of different genotype are recorded and analysis is then explicitly provided for each genotype, and not only as average values.

Identification of individual flies may even improve the accuracy of event detection, especially of the bottom-up approach introduced in 5.2.5.1, experience has shown that a set of input observables sorted by a fly identifier, such that attributes of individual flies are always provided in the same order, leads to an accuracy boost from 90 % to 95 % of correct classifications within the same test set. Accordingly, fly identities carry a valuable information content.

Besides all these technical reasons the identification of individual flies is particularly important for many users and was therefore considered to be essential for the system's acceptance within the field.

5.2.1.1 Occluded and Non-occluded Sequences

The postprocessor automatically discriminates between *occluded* and *non-occluded* sequences and handles them accordingly (see e.g. figure 30 p. 76). Here a short definition of two types of sequences. Intuitively, in case a frame contains two body areas it is considered as non-occluded, even if wing areas are merged, in case the two fly body areas are merged and a single connected region was detected instead, the frame is considered as occluded. Further, frames containing two body regions where one region is less than ten percent of the other are also considered as occluded.

Formally the number of connected regions condition is easily expressible utilizing the morphological erode-to-point operation $e^{\intercal}$.

Definition 34 *Let R be a binary region and $e(R)$ the morphological erosion operation. Further let $e^0(R) = R$ and $e^{i+1}(R) = e(e^i(R))$. The operation erode-to-point $e^{\intercal}(R)$ is defined as the least fix point with $e^{\intercal}(R) = e^{\intercal+1}(R)$.*

Definition 35 *Let B contain the body region containing up to two body areas of frame f, $|R|$ denote the number of active pixels in binary region R and $e^{\intercal}(R)$ the morphological erode-to-point operation. Further, let $Area_{small}$ denote the number of pixels of the smaller the smaller body area and $Area_{big}$ the number of pixels of the bigger body area and d denote dirt threshold, by default $d = 0.1$. Frame f is considered as occluded in case $|e^{\intercal}(B)| = 1$ or in case $\frac{Area_{small}}{Area_{big}} \leq d$.*

Conclusion 36 *Frame f is considered as non-occluded in case $|e^{\intercal}(B)| = 2 and \frac{Area_{small}}{Area_{big}} > d$.*

Such non-occluded sequences are called σ sequences, occluded sequences are called τ sequences. The sequence of all video frames V may be partitioned in alternating σ and τ sequences. Formally this is defined as follows.

Definition 37 *Let f be a frame. Function $o(f)$ is defined as $o(f) = 1$ in case f is occluded and $o(f) = 0$ otherwise.*

Definition 38 *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 $\forall f_i \in \tau : o(f_i) = 1$.*

Definition 39 *Let V be a sequence of frames f_i, $f_i \in V$. The function $O(V)$ is defined as sequence of values Ω with $\Omega_i = o(f_i)$.*

Conclusion 40 *The gradients of Ω, $\Delta\Omega$, mark borders of a partitioning Φ of V that consists of alternating σ and τ sequences.*

As a consequence of the definitions above, the set of all σ sequences Σ and the set of all τ sequences Υ therefore refer to all odd and all even sequences of such a partitioning Φ, depending on whether the first frame is an occluded or a non-occluded frame. For every frame it may easily computed to which of the ordered sequences it belongs, of course σ frames always belong to σ sequences and τ frames always to τ sequences.

Conclusion 41 *Let Φ_{odd} and Φ_{even} be subsets of Φ with with odd indices, $\Phi_{odd} = \{\phi_k | \bmod(k, 2) = 1\}$, and $\Phi_{even} = \{\phi_k | \bmod(k, 2) = 0\}$. The set of all σ sequences in Φ, denoted as Σ, and the set of all τ sequences in Φ, denoted as Υ, are:$\Sigma = \Phi_{even}, \Upsilon = \Phi_{odd}$ in case $o(f_1) = 0$ and vice versa otherwise.*

Conclusion 42 *For a frame $f_i \in V$ the sum $k = \sum_{j=1}^{i} \Delta\Omega_j$ indicates that f_i is in the kth subsequence of the partitioning Φ, $f_i \in \phi_k, \phi_k \in \Phi$. And trivially, in case $f_i \in \sigma$ it follows that $\phi_k \in \Sigma$.and $f_i \in \tau$ implies that $\phi_k \in \Upsilon$.*

Conclusion 43 *Let Φ be a partitioning of V and $\preceq$ denote the temporal order of frames and sequences, such that $a \preceq b$ denotes a occurs before b. By appending two artificial occluded frames at the very beginning and the very end of video V, f_0 and $f_{|V|+1}$, the partitioning in Φ is guaranteed to start and end with a τ sequence and that the order of sequences in Φ with $\phi_k \in \Sigma$.and $\varphi_k \in \Upsilon$ always is $\varphi_k \preceq \phi_k \preceq \varphi_{k+1}$.*

Notation 44 *A video V with additional frames f_0 and $f_{|V|+1}$, such that $f_0 \preceq V \preceq f_{|V|+1}$, is denoted as V'.*

The partitioning Φ of video frames V' and its structuring of frames in alternating σ and τ sequences is particularly useful when attacking the occlusion problem in section 5.2.2, especially for formulation of the SDH-Algorithm (p. 5.2.2.2 in section 5.2.2.2). The rest of this section explains how fly identities are assigned for non-occluded sequences in Σ and for occluded sequences in Υ.

5.2.1.2 Identity Assignment for σ Sequences

A frame that contains two body regions is considered as non-occluded, even if the wing areas are merged, and fly trajectories can be reliably tracked across sequences of non-occluded frames using the Hungarian algorithm [17]. The Hungarian algorithm is a combinatorial optimization algorithm that solves the assignment problem and is applicable to compute minimum cost assignments for bipartite graphs. The flies within two consecutive frames are therefore modeled as nodes and edges connect flies from one frame to all flies to the other frame, further the distances between flies are assigned as costs for corresponding edges. The computed minimum cost assignment is directly interpretable as an assignment of flies within the two frames.

Example 45 *Let $G = (V, E)$ be a bipartite graph with flies of two consecutive σ frames as nodes and edges connecting flies from one frame to all flies to the other frame. Since σ frames are known to have exactly two body areas, here denoted as A and B for the first and X and Y for the consecutive frame, the graph is defined as $V = \{A, B, X, Y\}$ with bipartite nodes $V_1 = \{A, B\}$ and $V_2 = \{X, Y\}$, $V_1 \cup V_2 = V$. Edges are therefore defined as $E = \{AX, AY, BX, BY\}$, an edge NM denotes a connection between nodes M and N. Further a distance matrix D is specified as $D = \{d_{NM}\}$ such that every edge $NM \in E$ contributes a cost value d_{NM} equal to the euclidean distance between the Centroids of regions N and M. The result of the Hungarian algorithm $F = Hungarian(G, D)$ is defined to deliver a perfect matching $F \subseteq E$, which is in this example, dependent on the values in D, either $F_1 = \{AX, BY\}$ or $F_2 = \{AY, BX\}$. In case $F = F_1$ the fly body A of the first frame is identified as fly body X in the consecutive frame and body B is identified as body Y, in case $F = F_2$ bodies are identified vice versa, A to Y and B to X.*

Notation 46 *The function utilizing the Hungarian Algorithm to assign bodies in frame f_i to f_{i+1} may be written as $F = Hungarian'(f_i, f_{i+1})$, its result may be abbreviated as $F = 0$ in case the first node in f_i is the first node in f_{i+1} or $F = 1$ for the other case.*

The first frame f_1 of every σ sequence is initialized with identifier values, such that the fly object containing the *left* body region[47] in $f_1^k \in \sigma_k$ is assigned with an identifier of value $a = 2k + 1$, the fly object containing the *right* body area is assigned with identifier $b = 2k + 2$. The identifier values a and b are then passed through consecutive frames and until the end of the sequence $f_{|\sigma_k|}^k$ is reached, flies with the same identifier are identified as the same flies within that sequence.

It is of course desirable to assign fly identifiers across different sequences, however, the Hungarian algorithm is not applicable for τ sequences as they only contain one body region. In principle, the Hungarian algorithm may be used to assign the last frame of sequence σ_k to the first frame of sequence $\sigma_{k=1}$ as $F = Hungarian'(f_{|\sigma_k|}^k, f_1^{k+1})$ (see method *pos* in section 5.2.2.1), however, this is only one possible approach, section 5.2.2 deals with the inter-sequence identity assignment problem, where flies identifiers (a_k, b_k) are matched with identifiers (a_{k+1}, b_{k+1}).

The system first determines identities of non-occluded sequences σ_k and then automatically assigns the different sequence identifiers (a_k, b_k) to each other.

5.2.1.3 Identity Assignment for τ Sequences

For identities assignment within τ sequences a number of downstream processing steps is required. At first identities in non-occluded sequences σ_k (see section 5.2.1.2 above) are assigned. The identifier ordered attribute sets is then used to interpolate primary attribute values for enclosed τ sequences (see section 5.2.4 for *both* possible inter-sequence identifier assignments, the attribute sets for τ sequences therefore contain two *possible worlds (see section 5.2.4, figure 48)*. Further, the identifier ordered attribute sets and the perimeter of body regions are used to derive the most plausible inter-sequence identifier assignment $oidMap$ throughout the entire video (see section 5.2.2, in particular 5.2.2.2. When merging the interpolated worlds according to the $oidMap$ assignments and therefore selecting a specific τ sequence interpolation the identities within the τ sequence are implicitly given by the structure of the data within the selected world. This results from a pre-sorting of the attribute sets according to fly identifiers in σ sequences before the interpolation step and the swapping of fly data within these sets in case propagated $oidMap$ identifiers are equivalent to $F = 1$.

[47] The left body region is determined by comparing the body regions *Centroid* values, in case of same x component the upper region is chosen.

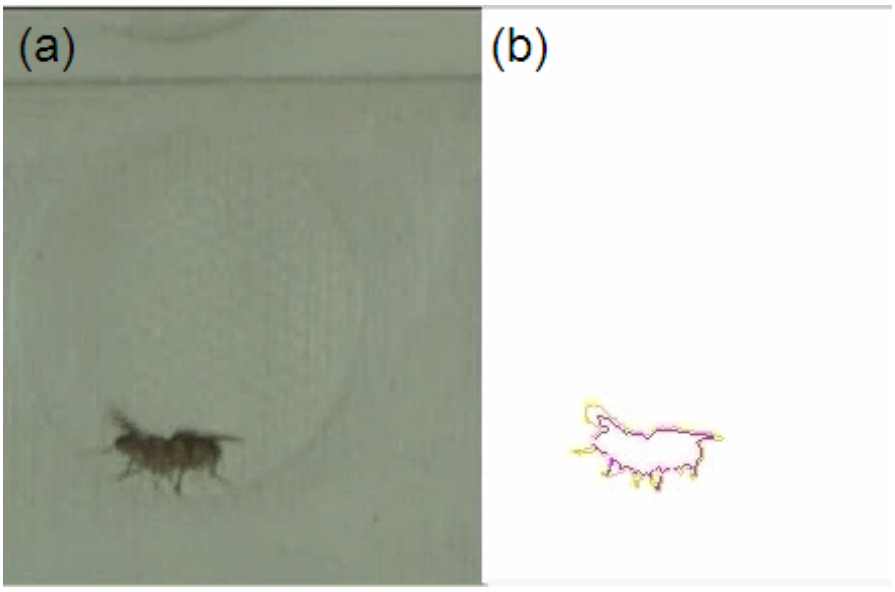


Figure 31: Occlusion sample. (a) original video frame depicting two overlapping flies. (b) extracted body region of the merged bodies in magenta, wing region in yellow.

5.2.2 Resolving Occlusions

When examining social interactions the aim is to monitor behaviour, especially when the individuals are close to each other. Therefore, in order to maintain automated tracking for high throughput and robust scores it is necessary to identify individual flies throughout the entire video even if they overlap or occlude each other from time to time.

If the two flies move close together and their body regions overlap each other correctly assigning the fly identities is a difficult task for a computer. Even for the human eye it may be sometimes difficult or even impossible to allocate individuals correctly after two flies have overlapped completely.

Figure 31 shows a sample occlusion. When the two flies overlap resp. occlude each other, the segmentation method detects only a single body region for both flies. This is a problem since it is highly desirable to identify individual flies throughout the entire video.

The aim of the next sections is to introduce and compare different strategies that attack the occlusions problem and to describe the basis and the history

that lead to the SDH-algorithm, which is the final method that is used by the system to solve the occlusion problem.

The structure of the section reflects the development path to the SDH-algorithm. There are two major approaches, within each of the approaches there are different classes of methods and within each of the classes the individual methods are described and evaluated in detail. The section starts with the first approach, the combination of t-methods, which is summarizing methods that are intuitive and "simple" and found the basis for the final solution. The second approach, the combination of s and t methods, introduces the idea of reasoning with paraconsistent information given in the videos. The SDH-algorithm is presented at the end of the section and introduces a combination of several methods. The idea of the algorithm is to deliver the global most plausible assignment of fly identities, it has an accuracy of 99.99 %.

5.2.2.1 Approach I: Combination of t-Methods

All methods that are used and evaluated within this first approach are classified as so-called *t-methods*.

A t-method is associated with τ sequences and aims to provide an assignment for the identifiers (a_k, b_k) and (a_{k+1}, b_{k+1}) (for definition see section 5.2.1.2) of its enclosing σ sequences. Furthermore many t-methods compute a *t-value* resp. a *t-score* for each assignment that resembles how sure a particular t-method was about the returned assignment.

Figure 32 gives an overview about a variety of different methods and classes of methods, this document introduces only a few methods for didactic reasons and the methods that made were finally incorporated into the system, in particular size-based methods and variants of the method *boc* turned out to be very useful.

In general, t-methods may be differentiated into ones following a "merge and split" approach and ones following a "straight through" approach, all methods used in this section may be classified in within these two categories.

"Merge and split" methods typically aim to match preserved characteristics before and after the occlusion while the occlusion itself is mostly treated as a set of unobservable states. Such methods therefore operate with values of the enclosing σ sequences, where both flies were segmented and identified, and mainly ignore the values in τ sequences.

Controversially, "straight through" methods aim to carry characteristic information through the states of occlusion. All straight through approaches may be adapted to fit into the template in algorithm 47 and therefore consist of

Figure 32: Overview of a large variety of t-methods, a few interesting point-based methods and area-based methods are introduced later in this section.

- a function $extract_{method}$ that extracts characteristic information $\mathfrak{C}_s$ (like e.g. a set of points) during occlusion state s.
- an function $partition_{method}$ that splits $\mathfrak{C}$ into a partition $\hat{\mathfrak{C}} = \{\mathfrak{C}^1, .., \mathfrak{C}^n\}$ according to track information T_s, such that the identifiers i for each tracked object $o_s^i, i \in \{1..n\}$, tracked in state s, are associated with non-overlapping subsets of $\mathfrak{C}$.
- a function $carry_{method}$ that takes $\mathfrak{C}_s$ from state s and carries $\hat{\mathfrak{C}}$ from state $s-1$ to state s,

Further, *all* methods should provide

- a function map_{method} that maps pre-occlusion-information to post-occlusion-information, either directly (merge and split) or according to the characteristic information carried over (straight through).
- a function $score_{method}$ that provides either a score $v \in (-1, +1)$ that resembles the quality (i.e. the certainty) of the method or NaN[48] if no conclusions may be drawn. NaN-values are equivalent to scores with value 0. Decisions with certainty 0 are treated as pure guesses and assumed to solve 50 % of the cases correctly.
- a function $resolve_{method}$ that provides a mapping I where it assigns pre-occlusion-identifiers (a_k, b_k) to post-occlusion identifiers (a_{k+1}, b_{k+1}) according to the value given by $score_{method}$ A simple but often useful implementation for this is $resolve_{thresh}$, depicted in algorithm 48 on p. 90.

Algorithm 47 ***Template for a straight-through strategy***

Function method(b, a)
  before occlusion, in state b:
    $\mathfrak{C}_b := extract_{method}(b)$
    $\hat{\mathfrak{C}}_b := partition_{method}(\mathfrak{C}_b, T_b)$
  during occlusion:
    for $s = b + 1$ *to* a
      $\mathfrak{C}_s := extract_{method}(s)$
      $\hat{\mathfrak{C}}_s := carry_{method}(\mathfrak{C}_s, \hat{\mathfrak{C}}_{s-1})$
    end-for
  after occlusion, in state a:
    $\mathfrak{C}_{a'} := extract_{method}(a)$
    $\hat{\mathfrak{C}}_{a'} := partition_{method}(\mathfrak{C}_{a'}, T_a)$
    $M_{b \to a} := map_{method}(\hat{\mathfrak{C}}_a, \hat{\mathfrak{C}}_{a'})$
    $v_{b \to a} := score_{method}(M_{b \to a})$
    $I_{b \to a} := resolve_{method}(v_{b \to a})$
    return $I_{b \to a}$
end function

[48] Abbreviation for "Not a Number".

Algorithm 47 provides a template for straight through strategies. The aim of a straight through method 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 some characteristics $\mathfrak{C}$ are extracted before and and after the occlusion and partitioned between the two detected flies, such that the partitioned characteristics $\hat{\mathfrak{C}}$ are associated with the identifiers of the detected flies. Then, for each frame during the occlusion, the characteristics is extracted and the associated identifiers are carried over from the characteristics of its predecessor frame. At the end, the characteristics carried through the occlusion $\hat{\mathfrak{C}}_a$ is compared with the freshly partitioned characteristics $\hat{\mathfrak{C}}_{a'}$, and a mapping, that indicates how the associated identifiers of the characteristics in $\hat{\mathfrak{C}}_a$ is match with the characteristics in $\hat{\mathfrak{C}}_{a'}$. The particular values of this mapping may compute a score that indicates how sure the method is about and its upcoming assignment result. According to that score the result assignment is derived and returned.

Point based methods

All point based methods follow the straight-through approach, in which a characteristic set of points $\mathfrak{C}$ that is traced "straight through" the occlusion states. The objects perimeter turned out to be a good choice for $\mathfrak{C}$, it outperformed all other tested sets for keypoint $\mathfrak{C}$ candidates either by solution quality or by computation time.

The evaluation process included a wide variety of point based methods, with $\mathfrak{C}$ extracted from appearance-, shape- or structure depending features. The results showed the perimeter as the feature contributing the most information content. The point based method where the characteristic point set is defined as the fly perimeter was therefore chosen as starting point for further variation and optimization steps. Further tests indicated varying distance metrics, filtering or filling the perimeter region do not improve the method's results, but that a simple dilation of the perimeter, a centroid correction and/or a concavity analysis of the perimeter itself may be simple extensions to a naive perimeter-carrying approach.

However, none of the optimization steps was convincingly able to balance the intrinsic problem coming with the voronoi-carrier $carry_{boc}$ (defined on p. 87), the problem is discussed in the "known weaknesses" paragraph on p..90

Still, the introduced point based methods within this section all correctly solve between 90 and 95 % of the test cases[49].

[49] The test cases contain all occlusions from a sample set of videos and provide a ground truth for more than 8000 occlusion cases.

KeypointBorder (boc)

This method was the first simple strategy developed, it was utilized to develop and test the strategy evaluation framework itself. It appears that no other point-set or metric delivers better results than this one. This method is therefore something like the "lowest common denominator" for all "useful" point-based approaches.

Implementation:

In order to fully characterize a method it is sufficient to define the functions used in the straight through template algorithm (see algorithm 47), for method *boc* these functions are defined as follows:

extract$_{boc}$: delivers $\mathfrak{C}$ as the border (i.e. the perimeter) of an object or a group of objects (see figure 33). This calculated by subtracting the once-eroded image from the original image. Note that the body region perimeter is extracted as primary attribute and therefore available for all fly objects (see 4.2.4).

(a) (b)

Figure 33: (a) bwImage of the two regions before Occlusion. (b) extracted characteristic information $\mathfrak{C}$ for method *boc*

partition$_{boc}$: assigns tracked identifiers to pre- or post-occlusion regions $\mathfrak{R}^1$ and $\mathfrak{R}^2$ (see figure 34 (a)), and splits characteristic points $\mathfrak{C}$ (see figure 33 (b)) into two subsets $\mathfrak{C}^1$ and $\mathfrak{C}^2$ (see figure 34 (b)), such that each point $p \in \mathfrak{C}$ is assigned to $\mathfrak{C}^1$ iff its the closest point in $\mathfrak{R}^1 \cup \mathfrak{R}^2$ lies in $\mathfrak{R}^1$ (and vice versa for $\mathfrak{C}^2$

resp. $\mathfrak{R}^2$). The closest point is looked up in a voronoi diagram[50][51], generated out of all points of $\mathfrak{R}^1 \cup \mathfrak{R}^2$ (see figure 35 (a)).

(a) (b)

Figure 34: (a) Two pre-occlusion regions $\mathfrak{R}^1$ and $\mathfrak{R}^2$ corresponding to figure 33a, colored according to known tracking information. (b) partition of $\mathfrak{C}$, according to identifier regions depicted in (a)

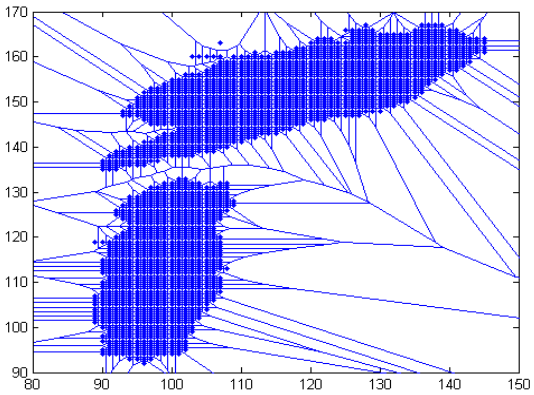
carry$_{boc}$: partitions point set $\mathfrak{C}_s$ from state s w.r.t. to the partitioning $\hat{\mathfrak{C}}_{s-1}$ of the previous frame. For this reason again a voronoi diagram[2] is generated out of the points of $\mathfrak{C}^1_{s-1} \cup \mathfrak{C}^2_{s-1}$ (see figure 35 (b)). such that each point $p_s \in \mathfrak{C}_s$ is assigned to $\mathfrak{C}^1_s$ iff $p_s \in V(p'_{s-1})$ and $p'_{s-1} \in \mathfrak{C}^1_{s-1}$ (resp. $\mathfrak{C}^2_s \ni p_s$ for $p'_{s-1} \in \mathfrak{C}^2_{s-1}$). Points with equal distance to two or more voronoi centers, e.g. lying on a blue line in figure 35, that further belong to different partition subsets in $\hat{\mathfrak{C}}_{s-1}$ are eliminated[52]. The iterative calls of $carry_{boc}$ are aborted when its last call resulted in a trivial partition $\mathfrak{C}^1_{s-1} = \mathfrak{C}_{s-1}$, $\mathfrak{C}^2_{s-1} = \mathfrak{C}_{s-1}$ or $\mathfrak{C}^1_{s-1} \cup \mathfrak{C}^2_{s-1} = \emptyset$, resp., in shorter notation, when $\exists i \leq |\hat{\mathfrak{C}}_{s-1}| : \mathfrak{C}^i_{s-1} = \emptyset$.

Figure 36 depicts how the partitioning derived in figure 34 (b) is carried through 5 consecutive frames.

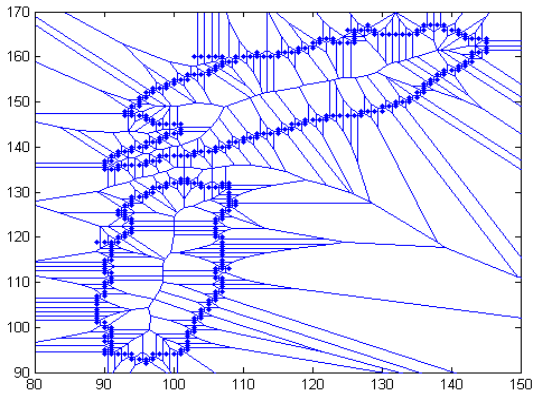
[50] Given a set of points S, a voronoi diagram $V(S)$ for S is the partition of a metric space (in our case a two-dimensional plane with euclidean distance $dist$) that associates a voronoi region $V(p)$ with each point (voronoi center) $p \in S$ such that $\forall v \in V(p), \forall p' \in S, p' \neq p : dist(v,p) < dist(v,p')$, i.e. such that all points in $V(p)$ are closer to p than to any other point from S.

[51] Voronoi diagrams are used since they are about 50-100 times faster than a naive comparison of minimum distances between points. Voronoi diagrams run in almost linear time $\mathcal{O}(n.logn)$, while naive methods usually run in $\mathcal{O}(n^2)$, i.e. the more points involved, the more voronoi diagrams outperform naive methods.

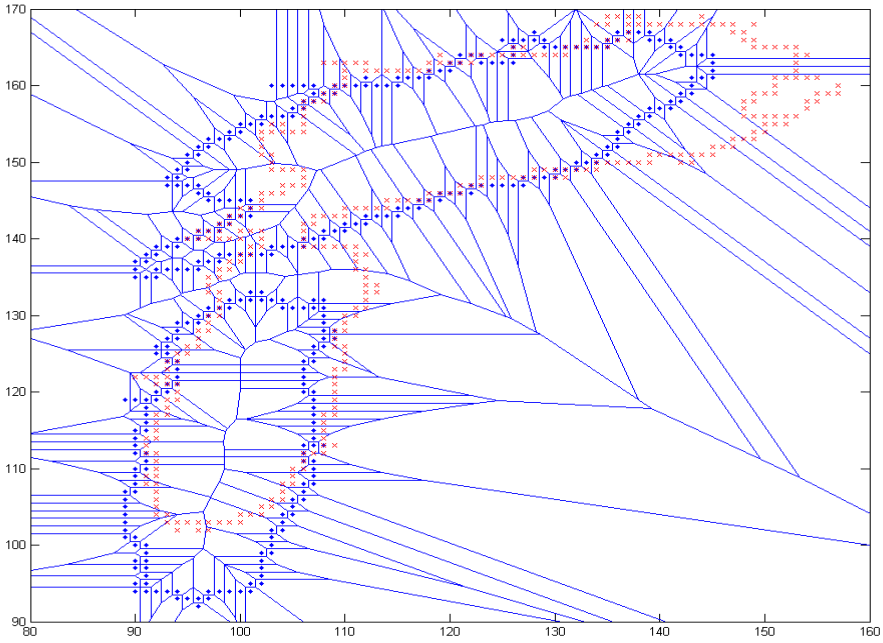
[52] Technically these points are assigned to a subset $\mathfrak{C}^3_s$ which will not be regarded in state $s+1$



(a)

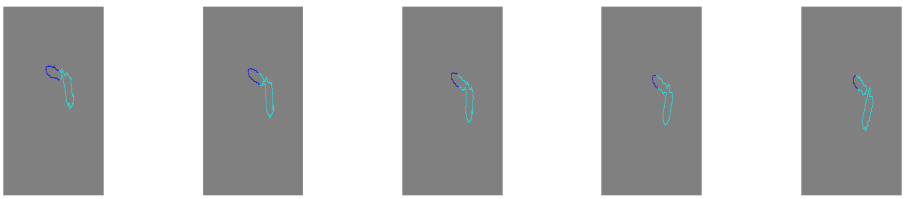


(b)



(c)

Figure 35: (a) Voronoi diagram for filled fly regions. Each point located in a region will take over the identifier value of the regions voronoi center (b) Voronoi diagram for characteristic information $\mathfrak{C}$ (c) Points of $\mathfrak{C}_{s+1}$ in regions of $V(\mathfrak{C}_s)$



(a) (b) (c) (d) (e)

Figure 36: (a)-(e): partitions of characteristic information carried through successive frames 226-230 in sample video 1

map$_{boc}$: compares the carried-through information $\hat{\mathfrak{C}}_a$ with the local partition $\hat{\mathfrak{C}}_{a'}$ and returns a matrix $M_{b\to a}$ summing points $|\{p_{ij}|p_{ij} \in \mathfrak{C}_a^i, p_{ij} \in \mathfrak{C}_{a'}^j\}|$ for $i, j \in \{1, 2\}$, in particular

(1,1) the number of points in $\hat{\mathfrak{C}}_a^1$ that are closest to points of $\hat{\mathfrak{C}}_{a'}^1$, that are thus indicating a desirability that identifier o_b^1 should be mapped to o_a^1, noted $o_b^1 \rightharpoonup o_a^1$,

(1,2) the number of points in $\hat{\mathfrak{C}}_a^1$ that are closest to points of $\hat{\mathfrak{C}}_{a'}^2$, voting for $o_b^1 \rightharpoonup o_a^2$, resp.

(2,1) the number of points in $\hat{\mathfrak{C}}_a^2$ that are closest to points of $\hat{\mathfrak{C}}_{a'}^1$, noted $o_b^2 \rightharpoondown o_a^1$ and

(2,2) $o_b^2 \rightharpoondown o_a^2$.

A mapping of object ids may be derived straightforward when matrix $M_{b\to a}$ is known, functions $.score_{boc}$ and $resolve_{boc}$ below provide a useful implementation for that.

score$_{boc}$: is defined as $\frac{(o_b^1\rightharpoonup o_a^1)+(o_b^2\rightharpoondown o_a^2)-(o_b^1\rightharpoonup o_a^2)-(o_b^2\rightharpoondown o_a^1).}{(o_b^1\rightharpoonup o_a^1)+(o_b^2\rightharpoondown o_a^2)+(o_b^1\rightharpoonup o_a^2)+(o_b^2\rightharpoondown o_a^1).}$, in particular, the sum of votes for mapping identifier o_b^1 to o_a^1 and identifier o_b^2 to o_a^2 (i.e. the votes for "odd-odd") minus the votes for mapping o_b^1 to o_a^2 and o_b^2 to o_a^1 (the votes for "odd-even"), normalized by the sum of all votes, such that the result is guaranteed to be between -1 and $+1$.

resolve$_{boc}$: $resolve_{thresh}$ (see algorithm 48), improved thresholds $T_{odd-odd}$ and $T_{odd-even}$ may be derived empirically. All tests have been executed with trivial thresholds $T_{odd-odd} = T_{odd-even} = 0$, these default thresholds for each method turned out to be sufficient for downstream computation.

$M_{b\to a}$	o_a^1	o_a^2
o_b^1 →	38	0
o_b^2 →	68	201

(a) (b) (c)

Figure 37: (a) partition $\hat{\mathfrak{C}}_a$ of frame 230 carried through from pre-occlusion tracking information (b) partition $\hat{\mathfrak{C}}_{a'}$ of frame 230 out of post-occlusion tracking information (c) sample mapping matrix $M_{b\to a}$ from frame 225 to frame 230, $M_{225\to 230}$, containing votes (i.e. matched identifiers) for odd-odd resp. odd-even mapping

Algorithm 48 ***Sample resolve function***

Function $resolve_{thresh}(v, T_{odd-odd}, T_{odd-even})$
  if $v > T_{odd-odd}$
    return "odd-odd"
  elseif $v < T_{odd-odd}$
    return "odd-even"
  else
    return "don't know"
  end-if
end-function

According to this and algorithm 48 occlusions solutions are suggested by the signs of score values. However, the amplitudes of score values still contain useful information about the certainty of the suggested solution.

Known Weaknesses:

The weakness of point based methods essentially comes with the voronoi carrier, the voronoi carrier refers to the mechanism depicted in figure 35 (c),

where each identifier is carried through its successive occlusion frame by taking the identifier of the nearest pixel in the previous frame (see figures 35 and 38).

Due to the fact that each pixel takes over the identifier of its voronoi-nearest pixel in the previous frame, crossing flies (see figure 38d) are likely to be mis-scored. Although all identifiers are derived from its directly successive frame, taking the pre-occlusion frame for all further comparisons - obviously a step back - probably wouldn't make it much worse than it is, which leaves some room for improvements here. In fact all mis-scores (except one jump, that was treated as occlusion case by accident) and most "don't know" cases of *boc* result from this known issue.

Measured Results and Conclusion

The strategy evaluation framework generates, besides statistical summaries, result figures for all examined videos and strategies. Figures 5.2.2.1, 5.2.2.1 and 5.2.2.1 depict these result figures for a sample movie 1. Within these figures the results of an evaluated method is visualized, sometimes in the context of extracted *blob-features* of the occlusion. Such blob-features are a set of attributes that is derived from the merged occluded region, in particular the duration of the occlusion and the minimum number of blob-pixels, a simple characterization of the "degree of occlusion" turned out to be interesting.

Figure 5.2.2.1 depicts the duration of occlusions and the minimum area during occlusion for all correct, "don't know" and the wrong classified cases, it visualizes how the method performs on different "types" of occlusions.

Figure 5.2.2.1 depicts the sorted and wrong scores for these occlusions and gives an impression of the distribution of the test cases and about the reliability of the methods score values.

Figure 5.2.2.1 further provides a scatterplot where all important attributes and method values are drawn against each other, scatterplots are useful to give an overall impression of a method, similar scatterplots indicate related methods.

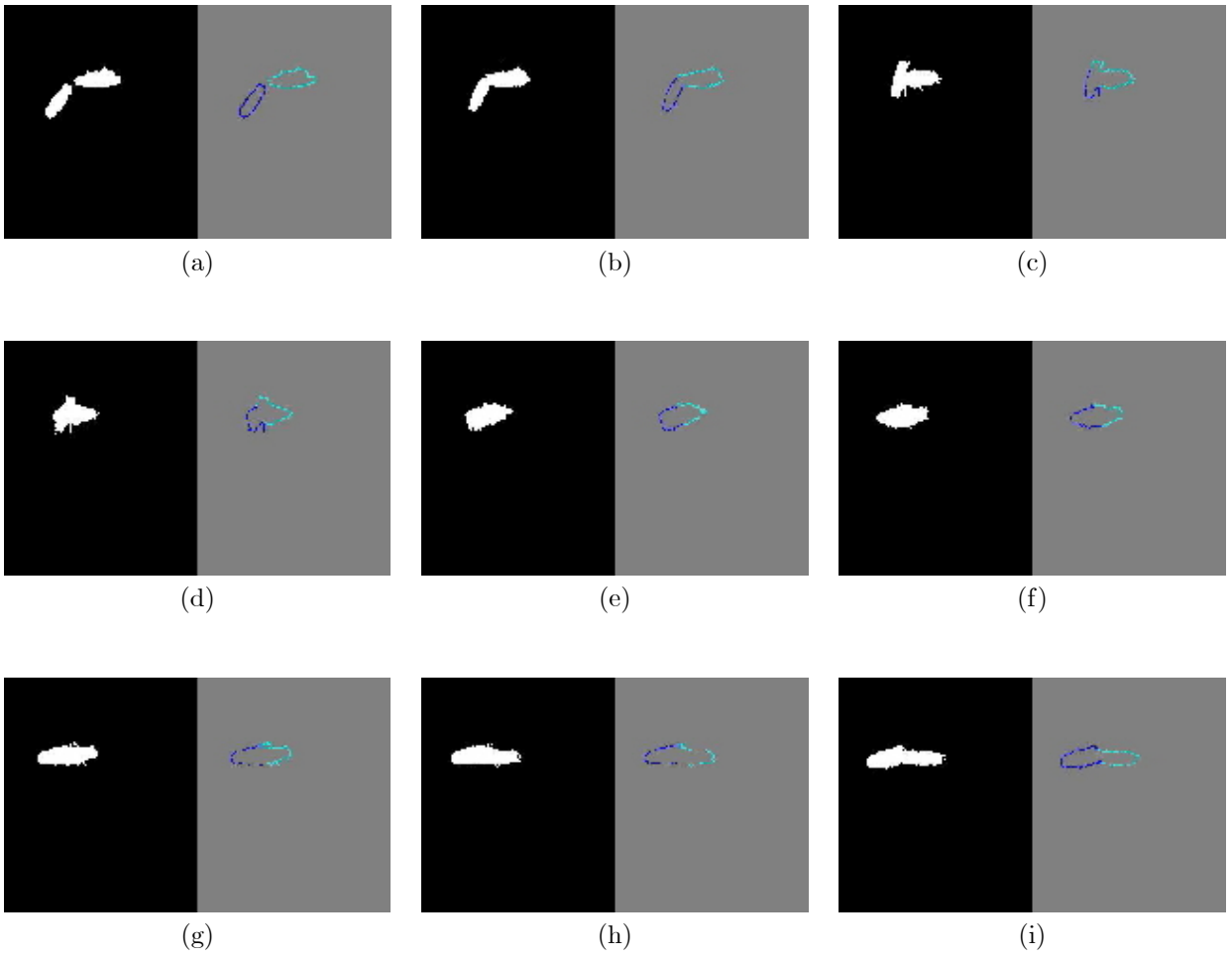
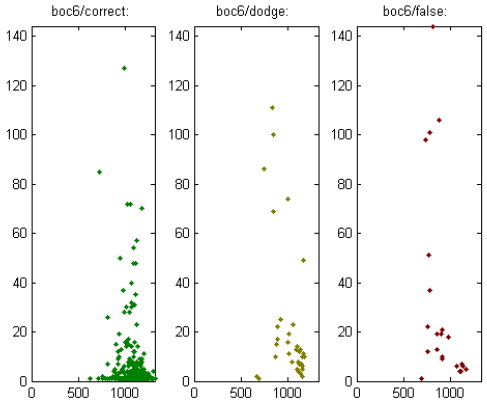
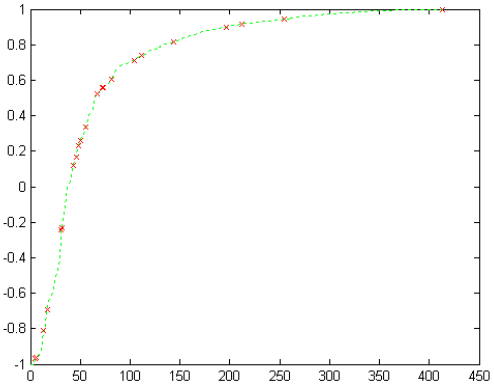


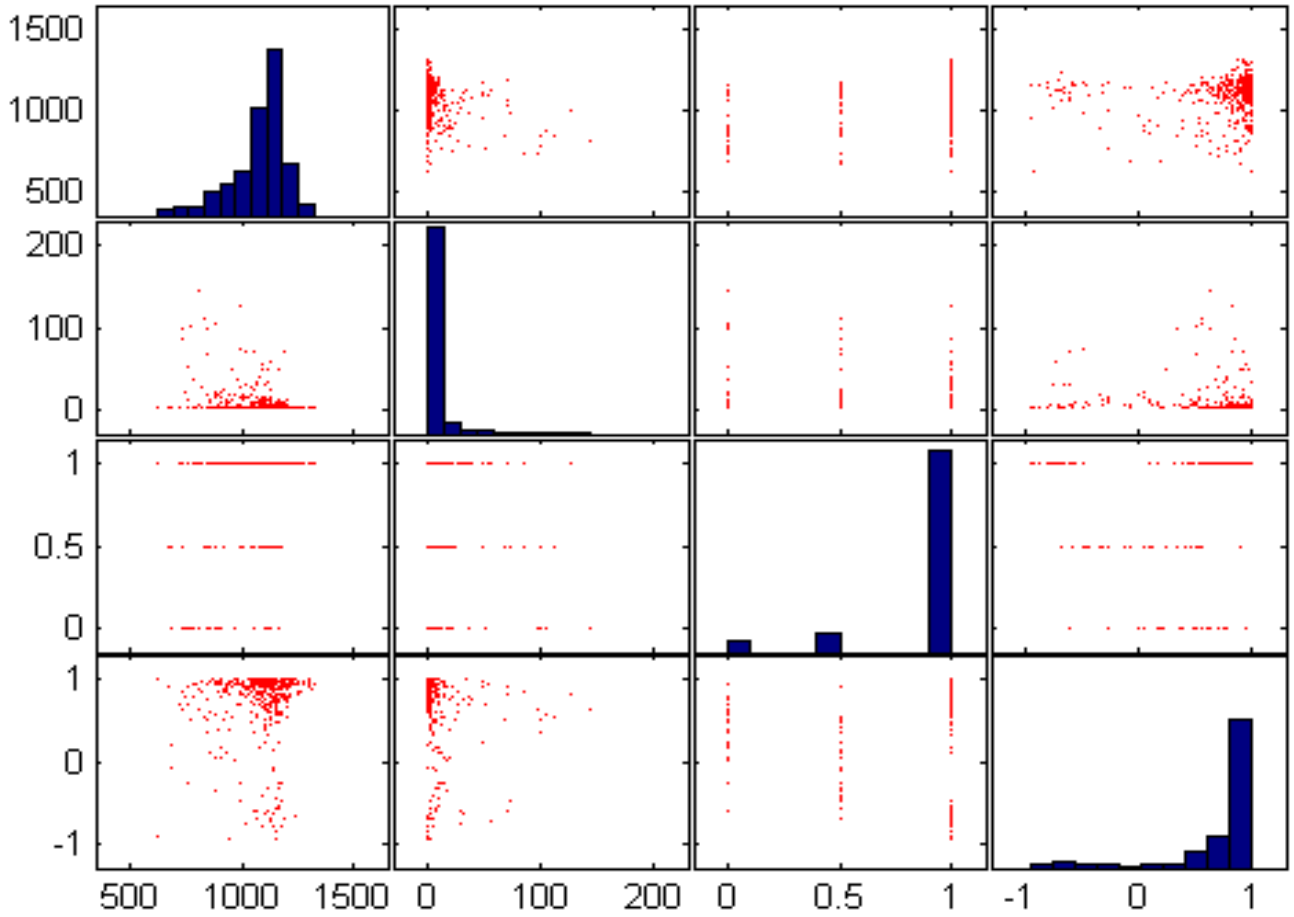
Figure 38: (a)-(i): boc output for video 1, frames 3077-3087. Although the flies are crossing boc method reports "no cross" here, frame (d) catches reason for this, intrinsically coming with the voronoi carrier



Scatterplot for boc method in video 1, the first subplot depicts the duration of occlusions (y-axis) vs. minimum area during occlusion (x-axis) for all correct cases. The other two subplots show the "don't know" and the wrong classified cases at the same scale.



depicts the (sorted) scores of video1 (green line) and marks cases wrongly classified by boc (red x's).



scatter-plots the attributes (1) minimum area [pixels], (2) duration [frames], (3) validity (1..correct, 0.5..don't know, 0 wrong) of boc result and (4) score for boc method for occlusion cases in video 1. The attributes are plotted columnwise at the x-axis and rowwise attributes at the y-axis against each other; the diagonal of the plot matrix depicts the value distribution for each attribute in a histograms.

The *boc* method has in principle **several ways for scoring "don't know"**:

- when a score is between the thresholds $T_{odd-odd}$ and $T_{odd-even}$, but as both values are set to 0 for this test series, this case doesn't occur in practice
- in other cases all keypoints end up with the same identifier, resulting in a "degenerated" mappings like e.g.

  100 points voting for $o_b^1 \rightharpoonup o_a^2$

  0 scores voting for $o_b^1 \rightharpoondown o_a^2$

  90 points voting for $o_b^2 \rightharpoonup o_a^1$

  0 points voting for $o_b^2 \rightharpoondown o_a^1$

  For *boc* the number of keypoints per objects, i.e. usually the *size*[53] of the objects, would turn the balance here, which I consider as too close to guessing. However, in most cases the male follows the female, thus usually male IDs are overwritten by female IDs (see figure 39a-i) and the final mapping would be right, as the female is usually the bigger one. However, since the occlusion assignment would depend on behavioural and anatomical features only the point based method refuses to score here. Anatomical features may be incorporated by attribute based methods (later introduced in this section), behavioural features is strictly not taken for occlusion assignments.
- sometimes the point-based methods might loose all points. Anyway, this shouldn't happen, especially not for the boc method.

If a keypoint-based strategy encounters such a case, it immediately stops all attempts and returns "no score". resp. "don't know". For video 1 (figure 5.2.2.1) all "don't know"-cases result either from crossings flies.or from flies identifiers being "overwritten" while following the other fly.

The observed wrong scored cases contained 19 crossing cases, one jump, that may be detected as jump separately, and one "unlucky cover" (see figure 40), where one flies covers the other just enough to wipe out its identifiers (see also "known weaknesses" and figure 38 below. The wrong score of 0.94 in figure 5.2.2.1 is depicted in figure 38.

The *boc* method seems to be good in non-crossing occlusion cases with one static fly. As soon as one fly follows the other it is endangered to suffer from the known voronoi-carrier weakness. The variant *boc*6 introduced below therefore tries to compensate such movements, which would reduce that the effect of the known weakness, but still leave some "uncompensatable" crossing cases like the one in figure 38.

[53] To be exact, its the number of pixels the perimeter consists of, which should heavily dependent on the objects size.

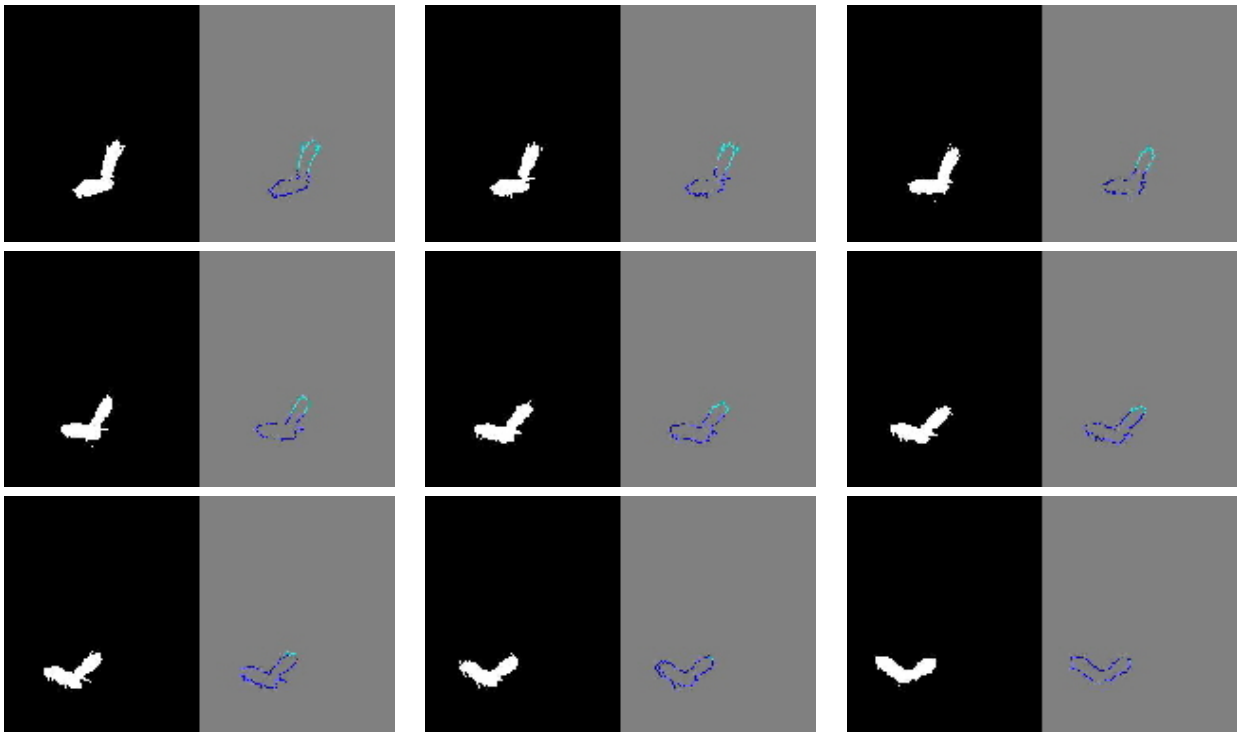


Figure 39: (a)-(i): boc output for video 1, frames 4592-4600. All identifiers of the male fly are overwritten by female identifiers, resulting in "don't know" for frame 4601.

Variant "rigid transformation" (boc6)

This variant tries to balance the known weaknesses of the voronoi-carrier (see figure 35 on page 88 above) by performing rigid transformations[54] between successive frames.

More particular, figure 35 shows that the red dots of the successive frame are misplaced due to the movement of the flies during occlusion. As e.g. one fly follows the other, more and more error accumulates, resulting in both flies being covered with identifiers of the "fleeing" fly only until the method finally can't take a decision and ends up with a score of 0.

The goal of this variant is to improve the *boc* method by "pulling back" the red dots in the voronoi region, such that the accumulating error is at least reduced. The first attempt was to apply a "rigid transformation", i.e. translation and rotation of the occluded region, in order to compensate observed

[54] The version implemented so far doesn't compensate rotation but goes for translation only. The difference to *boc*5 ("centroid correction", see supplementary) is that *boc*6 processes "fair" boundaries where "heavily-zig-zagging lines" or holes (i.e. inner boundaries around the holes) do not contribute more pixels and thus do not attract the center of gravity.

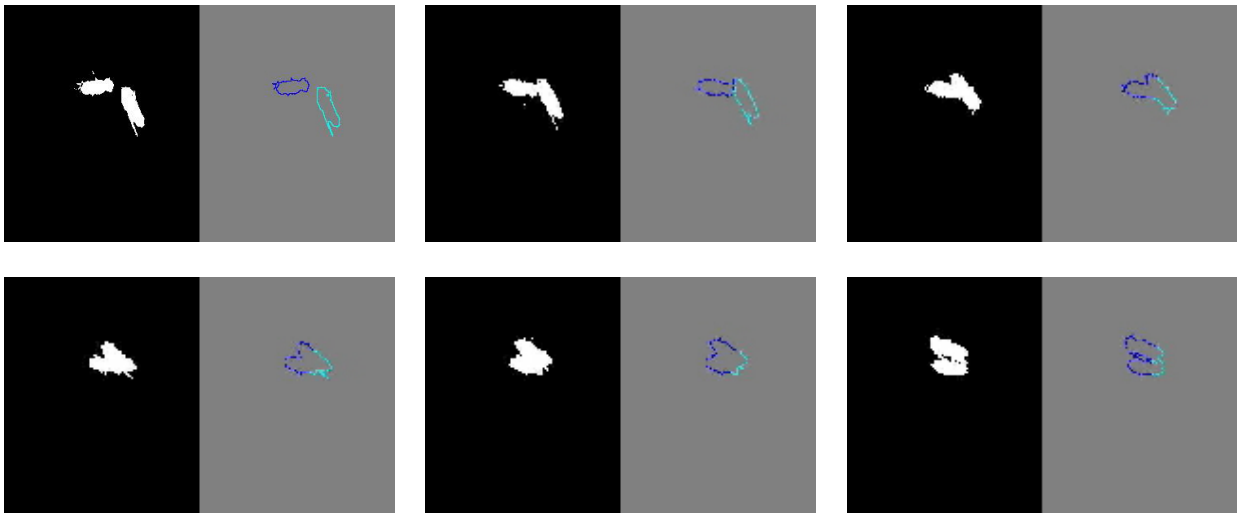


Figure 40: (a)-(i): boc output for video 1, frames 3240-3245. Although the flies are crossing boc method reports "no cross" here, frame (c) catches reason for this, intrinsically coming with the voronoi carrier (see also "known weaknesses" below)

movements, however, it turned out that compensation translations only is more reliable than compensating rotations as well, this property holds for non-rigid transformations[55] as well. The idea of the final method is simple, it corrects the coordinates of characteristic points p by the movement of the total occluded region before matching points within the voronoi diagram.

In particular the refinement is defined as follows:

Let S be a set of points and $G(S)$ be the center of gravity for S. Let $\mathfrak{R} = \mathfrak{R}^1 \cup \mathfrak{R}^2$ be the region that covers both flies, whether they are occluded or not. Let b be the last frame before occlusion and a the first frame after the occlusion. Algorithm 49 formally describes how the points in point sets $\mathfrak{C}_i$ are corrected in order to compensate movements of blob region $\mathfrak{R}$.

Algorithm 49 ***Translation Correction***

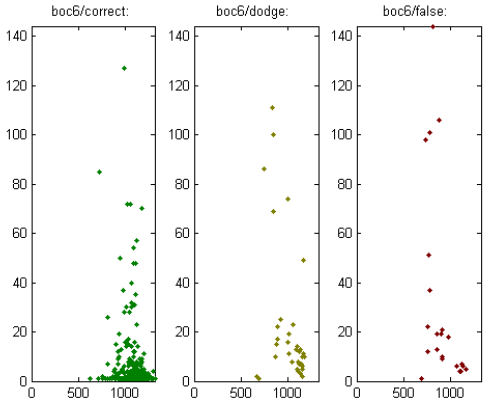
for each frame $i \in (b+1, a)$*:*
  for each point $p_i^j \in \mathfrak{C}_i$*:*
    $p_i^j := p_i^j + G(\mathfrak{R}_i) - G(\mathfrak{R}_{i-1})$
  end-for
end-for

[55] The tested non-rigid transformation also aim to compensate rotation as well, the idea of these approaches was inspired by *Procrustes Analysis*. The Procrustes Analysis step is used to align shapes e.g. for active shape model extraction, where shapes are aligned to an equivalence class and thus become comparable, such that a mean shape and variance of that shape is computable.

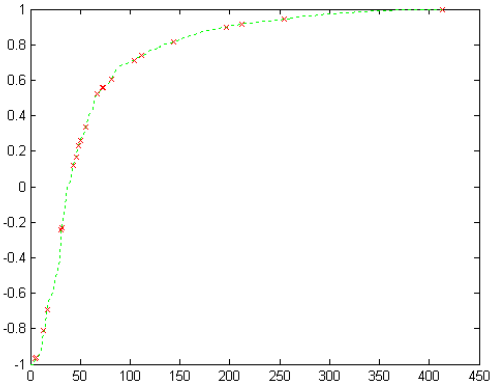
All further function are identical to the ones used for method *boc*.

Measured Results and Conclusion

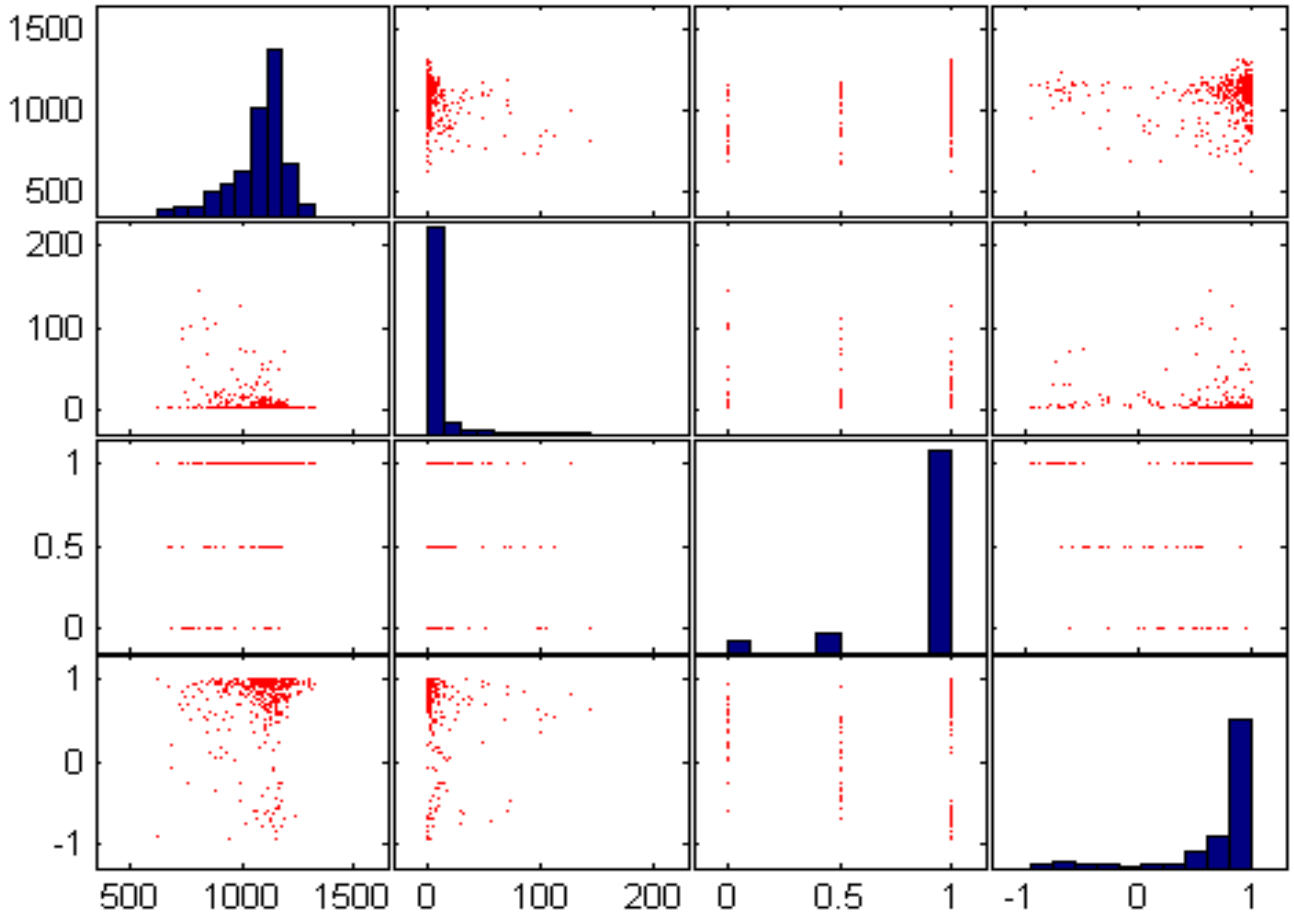
Figures 5.2.2.1, 5.2.2.1 and 5.2.2.1 depict the measured results as before, but with values returned by method *boc*6. The key idea is to compensate the translation of the merged region and thus improve the results of the voronoi carrier.



Scatterplot for boc6 method in video 1, the first subplot depicts the duration of occlusions (y-axis) vs. minimum area during occlusion (x-axis) for all correct cases. The other two subplots show the "don't know" and the wrong classified cases at the same scale.



depicts the (sorted) scores of video1 (green line) and marks cases wrongly classified by boc6 (red x's).



scatter-plots the attributes (1) minimum area [pixels], (2) duration [frames], (3) validity (1..correct, 0.5..don't know, 0 wrong) of boc6 result and (4) score for boc6 method for occlusion cases in video 1. The attributes are plotted columnwise at the x-axis and rowwise attributes at the y-axis against each other; the diagonal of the plot matrix depicts the value distribution for each attribute in a histograms.

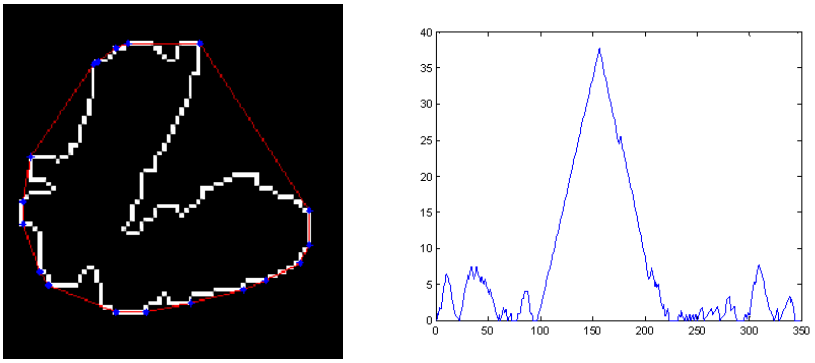
This method is again good for non-crossing cases and slightly outperforms *boc* for these cases. It attacks the problems coming with the known voronoi weakness, such as often observed occluded following flies that loose identifiers. The method works particularly well when the "constellation" of two flies remains the same, i.e. one fly behind the other when following, or one fly moving, the other orienting after it. The method again has its problems when flies cross or heavily overlap each other and when the general constellation heavily changes, e.g. before occlusion one fly is located after the other, during occlusion they swap places.

Variant "concavity-correction" (boc7)

This *boc* variant attempts to adjust identifier transitions between two characteristic points, identifiers are "shifted" such that the most concave point, which the inner most point, marks the new identifier transition. The idea is that two flies occluding each other in an angle (see figure 41 (a)) may be resolved by those inner most points, these points should mark the transition between the flies The distances of perimeter points to their corresponding convex hull fragment is used to determine the concavity of each perimeter point (see figure 41 (b)), a sample input and output of the algorithm, showing this variants improvement in this particular case, is depicted in figure 42.

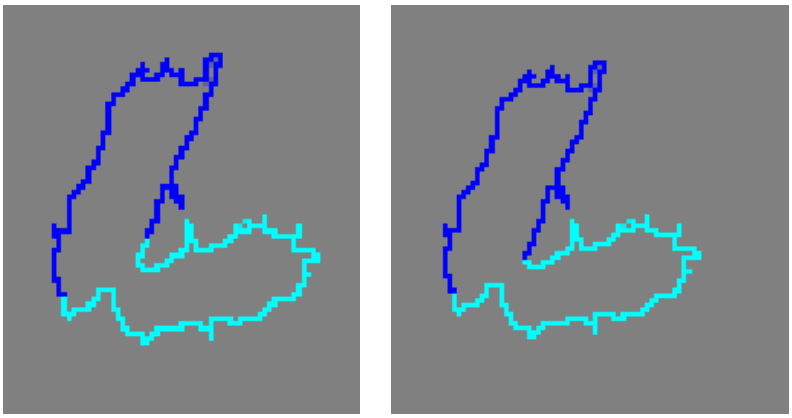
Measured Results and Conclusion

Figures 5.2.2.1, 5.2.2.1 and 5.2.2.1 depict the measured results as for the previous methods with values returned by method *boc*7.



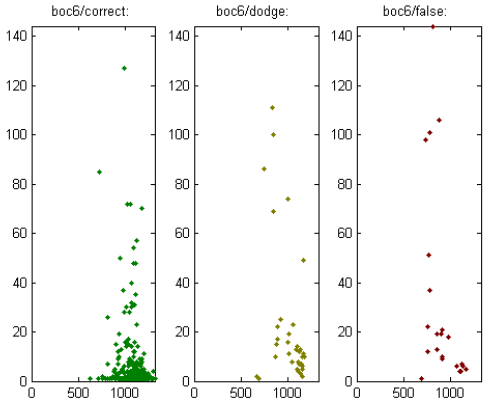
(a) (b)

Figure 41: (a) The white line shows the (smoothed) boundary of two occluded objects. The red line shows the boundary of the convex hull, which is build as a polygon of the blue points. (b) depicts the distance between the occluded objects perimeter (depicted in (a) as white line) to the boundary of the convex hull (depicted in (a) as red line), starting at the left-top point of the convex hull. The perimeter and the perimeter of the convex hull are fragmented between their shared points depicted in (b) as blue points), distances are measured between corresponding fragments and are computed by normals to the convex hull fragments.

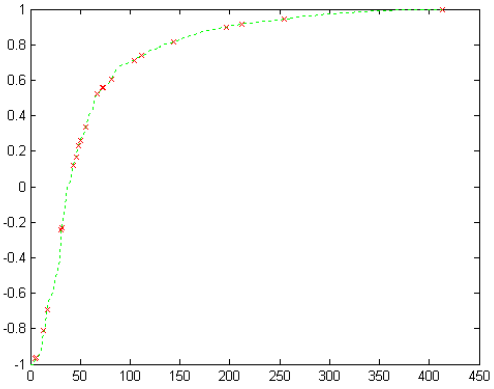


(a) (b)

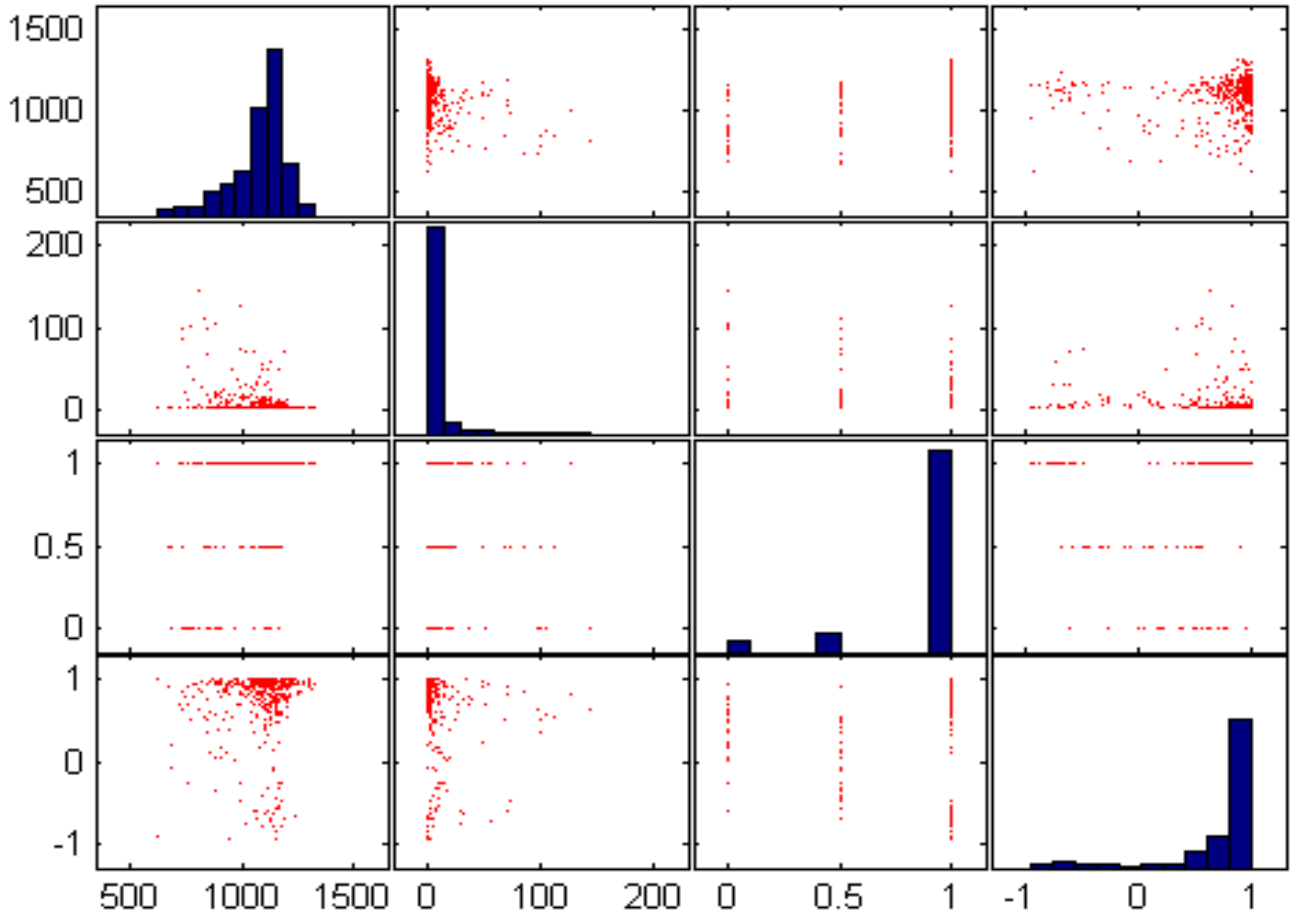
Figure 42: (a) sample input partition for the concavity improvement algorithm (b) output partition as algorithm result, where the transition between identifiers is adjusted according to highest peeks in figure 41 (b), as the "most concave" points are likely to be on the fly transition



Scatterplot for boc7 method in video 1, the first subplot depicts the duration of occlusions (y-axis) vs. minimum area during occlusion (x-axis) for all correct cases. The other two subplots show the "don't know" and the wrong classified cases at the same scale.



depicts the (sorted) scores of video1 (green line) and marks cases wrongly classified by boc7 (red x's).



scatter-plots the attributes (1) minimum area [pixels], (2) duration [frames], (3) validity (1..correct, 0.5..don't know, 0 wrong) of boc7 result and (4) score for boc7 method for occlusion cases in video 1. The attributes are plotted columnwise at the x-axis and rowwise attributes at the y-axis against each other; the diagonal of the plot matrix depicts the value distribution for each attribute in a histograms.

In cases where the two flies overlap in specific angles, such that the concavity measures in 41 (b) contains a clear peak value, the concavity corrected boc7 method outperforms *boc* and *boc*6. However, its overall performance is still similar to the methods before; still, when combining the results of different methods the scores and values computed by *boc*7 contribute new information content.

A further useful variant of *boc*7 might be to utilize the concavity-corrected borders (see 42 (b)), i.e. the shifted identifiers, as input for the competitive floodfill operation defined in section 4.2.3 (see definition 33 on p. 69).

Attribute based methods

These methods follow a classical merge and split approach, where the set of characteristic attributes, used to re-identify flies may vary. 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 the "better matching" post-occlusion flies. In principle any attribute can be taken into account for an attribute based method.

Attribute based methods still fit into the framework described in 5.2.2.1, p. 82:

- The functions $extract_{method}$ still extract characteristic information - this time the *attributes in non-occluded states*, the $partition_{method}$ functions assign the attribute-values to the individual flies. For attribute-based methods these functions are trivial, and mostly inherently given (see section 4.2.4).
- Further, there are no $carry_{method}$ functions - it would be a straight through approach if case such a function was required.
- The function map_{method} still maps pre-occlusion-information to post-occlusion-information, in most of the cases this map consists of distances $o_b^1 \rightharpoonup o_a^1, o_b^1 \rightharpoonup o_a^2, o_b^2 \rightharpoonup o_a^1$ and $o_b^2 \rightharpoonup o_a^2$ between attribute values o_b^1,o_b^2, o_a^1 and o_a^2 of flies 1 and 2, b denotes the last frame before and a the first frame after the occlusion.
- The function $score_{method}$ still provides a score $v \in (-1, +1)$, by default all scores are computed by $score_{default}$ as $(-1) * \frac{(o_b^1 \rightharpoonup o_a^1)+(o_b^2 \rightharpoonup o_a^2)-(o_b^1 \rightharpoonup o_a^2)-(o_b^2 \rightharpoonup o_a^1).}{(o_b^1 \rightharpoonup o_a^1)+(o_b^2 \rightharpoonup o_a^2)+(o_b^1 \rightharpoonup o_a^2)+(o_b^2 \rightharpoonup o_a^1).}$, i.e. the attribute distances between fly^1 to fly^1 and fly^2 to fly^2 minus the attribute distances between fly^1 to fly^2 and fly^2 to fly^1, normalized by the sum of all distances, such that the result is guaranteed to be between -1 and $+1$. As short distances values are typically preferred, the scores are scaled by (-1). Note that some attribute-based methods might not come with any meaningful score, but just a "decision" -1 or $+1$.

- By default, attribute-based method use the trivial function $resolve_{>0}$ which assigns fly^1 to fly^1 and fly^2 to fly^2 in case score $v > 0$.

Example: *position-based method (posm)*

Method pos compares fly positions, and pick the assignment with smaller sum of distances. This method follows the idea of applying the Hungarian Algorithm to assign the last frame of sequence σ_k to the first frame of sequence $\sigma_{k=1}$ as suggested in 5.2.1.2 p. 80, however, since the Hungarian Algorithm does not provide an applicable score, the sum of distances are used for a very similar assignment that provides a score.

$extract_{pos}$: extracts (x, y)-coordinates of the center of gravity of the fly regions.

map_{pos} : the resulting matrix $M_{b\to a}$ consists of the distances between these coordinates, such that it resembles

(1,1) distance between fly^1 before and fly^1 after occlusion, noted $o_b^1 \rightharpoonup o_a^1$,

(1,2) distance between fly^1 before and fly^2 after occlusion, noted $o_b^1 \rightharpoonup o_a^2$,

(2,1) distance between fly^2 before and fly^1 after occlusion, noted $o_b^2 \rightharpoonup o_a^1$,

(2,2) distance between fly^2 before and fly^2 after occlusion, noted $o_b^2 \rightharpoonup o_a^2$,

$M_{b\to a}$	o_a^1	o_a^2
$o_b^1 \rightharpoonup$	38	0
$o_b^2 \rightharpoonup$	68	201

A map $M_{b\to a}$ with sample values.

As *pos* uses the default scoring method, $score_{default}$, the score v of the sample values in figure 5.2.2.1 would be $(-1) * \frac{38+201-0-68.}{38+201-0-68.} = -0.557$.

size-based *method (siz2m,siz2)*

Size-based method match flies according to their size, the mean, maximum, minimum or any other aggregation of sizes before occlusion and after occlusion may be compared.

Technically all functions are defined as in method *pos* above, but **fly areas**, the number of pixels of the body regions, are taken into account instead of positions and distances.

In particular methods $siz2m$ and $siz2$ turned out to be useful, both methods compares the fly areas of the frame directly before the occlusion with the fly areas directly after occlusion. While $siz2m$ computes scores from size differences, method $siz2$ comes without scores. More particular.$siz2$ just assigns bigger flies to bigger flies, i.e. it checks if fly one has the bigger Area before occlusion $Area_b^1 > Area_b^2$, and after occlusion $Area_a^1 > Area_a^2$, and if so, it assigns fly_b^1 to. fly_a^1. Other combinations for $siz2$ are resolved straightforward:

$Area_b^1 > Area_b^2, Area_a^1 > Area_a^2, \Longrightarrow fly_b^1 \leftrightarrows fly_a^1, fly_b^2 \leftrightarrows fly_a^2$
$Area_b^1 > Area_b^2, Area_a^1 \leq Area_a^2, \Longrightarrow fly_b^1 \leftrightarrows fly_a^2, fly_b^2 \leftrightarrows fly_a^1$
$Area_b^1 \leq Area_b^2, Area_a^1 > Area_a^2, \Longrightarrow fly_b^1 \leftrightarrows fly_a^2, fly_b^2 \leftrightarrows fly_a^1$
$Area_b^1 \leq Area_b^2, Area_a^1 \leq Area_a^2, \Longrightarrow fly_b^1 \leftrightarrows fly_a^1, fly_b^2 \leftrightarrows fly_a^2$

Interestingly both methods, $siz2m$ and $siz2$, contribute information content and are therefore considered when combining different method results.

The intrinsic problem of using raw fly sizes as held in primary attribute $Area$ is that the methods performance is dependent on a size difference between the two recorded flies. Even when such a size difference is given, flies may turn up at the border region (see video frame in figure 43) such that the size values are undesirably distorted for these frames (see first data bar in 43). However, correcting of Area by observed Eccentricity values, $AreaE = Area\sqrt{1 - Eccentricity^2}$ defines a new attribute that captures the essence of the Area information also of up-turning flies in a robust way (see second data bar in 43). The desired effect is further strengthened when the incorporated eccentricity information is smoothed by a logistic function, $EccentricityC = \frac{Eccentricity}{1+e^{5 \cdot Eccentricity}}$ and $AreaEC = Area\sqrt{1 - EccentricityC^2}$ (see third data bar in 43). The example sequences before resp. after the occlusion (marked by the black box) depicted in figure 43 underline the experience that refined and robust size information in $AreaEC$ is a desirable improvement, e.g. for occlusion resolvment purposes or for frame validity determination (see section 5.2.3.1 p. 122). Therefore all advanced size-based methods incorporate advanced $AreaEC$ instead of simple $Area$ values.

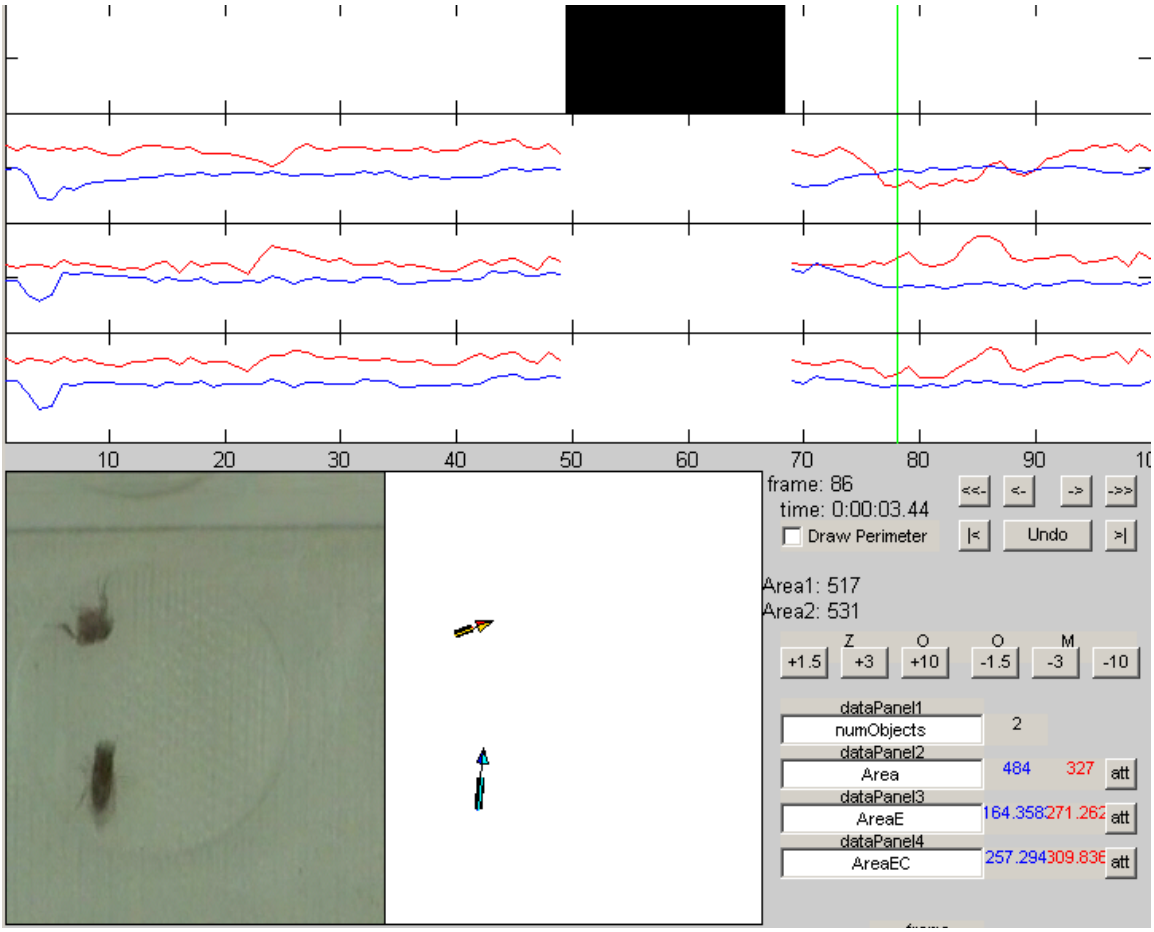


Figure 43: Snapshot of the annotationTool, depicting a frame where the usually bigger female fly "turns up" and therefore has smaller *Area* values. The snapshot further visualizes the values of attributes *Area*, *AreaE* and *AreaEC* over time and visualizes that the latter attributes successfully compensate up-turning flies and therefore provide more discriminative size values.

Combining Meta-Method and Conclusion of Approach I

After extraction of observable occlusion attributes, in particular the occlusion duration and its minimum number of pixels turned out to provide a good characterization, and after computation of all decision and score results for a large number of methods a *meta-method* may be derived using standard machine learning approaches. When evaluating different t-methods and meta-combination strategies the Classification And Regression Trees (CART) turned out to be a useful approach.

Meta-combination strategies are all derived during a machine learning phase, where for a limited number of cases all observable information, no matter where it was computed or derived, are provided together with a ground truth, the correct answers per case. During the learning phase a meta-method aims to derive the desired ground truth from the observable values. Further the information content of all input observables is analyzed, such that redundant or weak information that is not incorporated is eliminated. Such eliminated observables are not considered during later phases and there is no need to compute them any further.

A learned method is validated in a test phase, where the methods applies the rules it learned during the learning phase and derives decisions from given observables. This decisions may again be compared with ground truth values, in particular the cross-validation was used for method evaluation.

A sample CART decision tree, which incorporates the methods described in the paragraphs above, is depicted in figure 44. The derived decision rules may be read, e.g. for the right branch, as follows: In case method $siz2$ decides 1 and and method *bocd*, a variant of *boc* that operates on the dilated perimeter, agrees, then decide 1. In case bocd disagrees, consider the score of *boc* and the duration of the occlusion.

The figure should provide an impression about the principle of such trees resp. of meta-methods in general. Although alternative meta-method approaches performed equally well the tree approach was chosen because of its intuitive and easy understandable rule-based decisions.

The overall accuracy of the combined meta-methods within 99.9 % of correctly solved cases, a remarkable improvement since all underlying methods are only in 90 to 95 % of the cases correct. However, there is a significant disadvantage of this approach. When processing all cases linearly and assessing all occlusion sequences individually only one wrong identity assignment is passed through the entire video, in other words, the identities are swapped from that wrong assignment on and identities are therefore misassigned up to the end of the video. In order to overcome this disadvantage, which is intrinsic to all combined or non-combined t-methods, a completely new approach introduced in section 5.2.2.2, was required.

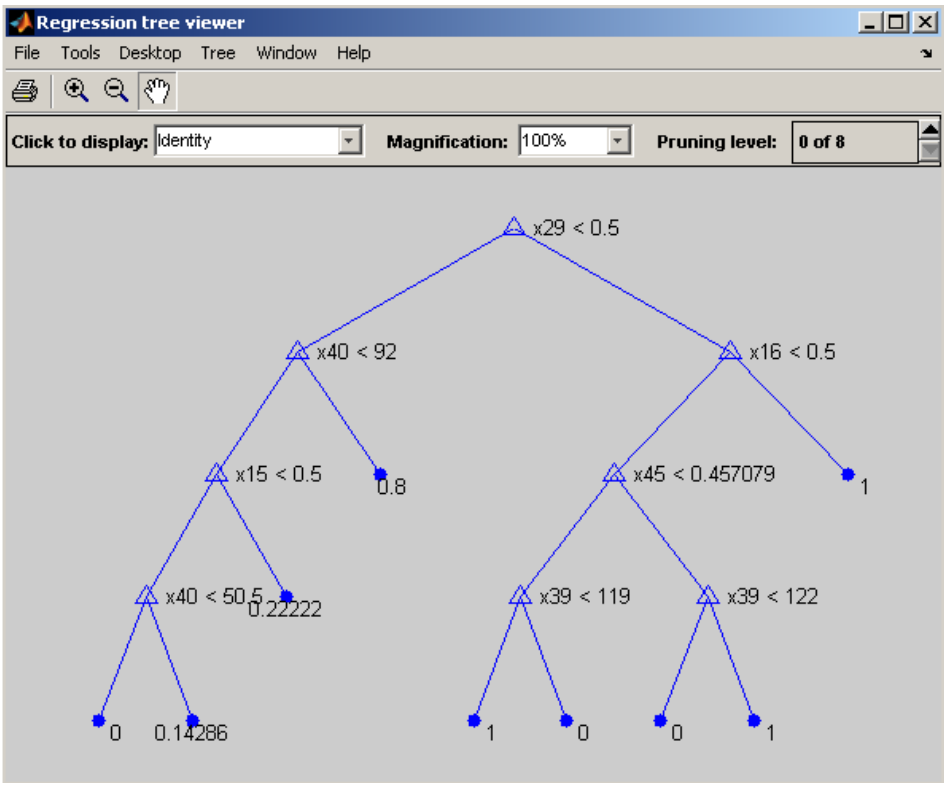


Figure 44: Classification And Regression Tree (CART) combining output of several t-methods and observable occlusion features.

5.2.2.2 Approach II: Combination of s and t-Methods

For the second approach t-methods, some examples were introduced in section 5.2.2.1, are complemented with so-called *s-methods*.

Section 5.2.1.1 introduced and σ and τ sequences and defined how a video may be partitioned into alternating occluded and non-occluded sequences σ and τ. While t-methods are associated with τ sequences, the newly introduced s-methods are associated with non-occluded σ sequences where two flies are detected, s-methods aim to discriminate and re-identify the two detected flies much like in the merge-and-split approaches introduced in section. 5.2.2.1. Contrary to merge-and-split t-methods, which assess and compare characteristics of the σ sequences directly before and after an occluded τ sequence only, extracted s-method characteristics require to be comparable during the whole video. Further, and again similar for t-methods, some s-methods may provide a *s-value* resp. a *s-score* for each assignment that resembles how sure the s-method was about the returned assignment.

This comparability property of s-methods opens up the possibility to overcome the intrinsic disadvantage of an approach that relies on t-methods only (see conclusion paragraph in section 5.2.2.1) and is essential for the overall approach described in this section.

The following two examples should further underline the difference between s and t methods.

The size-based method $siz2m$ (see section 5.2.2.1) aims to match flies before an occlusion in σ_b to flies after an occlusion in 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 non-occluded sequence σ_k may be matched to flies of every other non-occluded sequence σ_i, again such that the bigger and flies are assigned to each other.

On the contrary, the position-based method $posm$ (see again section 5.2.2.1), which aims to match flies before resp. after an occlusion according to their position, is *not* suitable for an s-method generalization. Obviously, the longer the time span between two sequences σ_k and σ_i resp. between two matched fly positions, the worse the expected results.

In general, anatomical features like size or eye color suggest suitable s-methods implementations. Intuitively, in case one fly had two heads while the other looks normal, it would be easy to discriminate the two flies by eye through the entire video. In principle any measurable anatomical or otherwise constant[56] feature that discriminates the two flies is applicable.

[56] Marking one fly, e.g. by painting a white dot on its back, as it is common in many biological experiment setups, gives also such a constant feature and therefore is, if reliably measurable, suitable for a s-method implementation.

A first intuitive combination of s and t-methods might be to select scores where a s-method is absolutely sure and to treat their corresponding identity assignments as "fix points", and then to use t-methods for low-score cases in between. Such an approach would limit the intrinsic problem coming with t-methods, wrong score would be propagated up to the next fix point rather than up to the end of the video. It is able to do so by exploiting paraconsistent information given by the s-method's underlying overall comparable feature.

The final approach introduced in this section is a further generalization of that idea.and enables that s-methods and t-methods **correct each other**.

SDH[57]-Algorithm

The SDH algorithm operates with s-values and t-values in order to derive the most possible occlusion assignment throughout the entire video.

In order to ensure that these values are comparable, s-values and t-values may be defined to be *probabilities* which are, in case not given directly from the method, derived from s-scores and t-scores, $-1 \leq score \leq +1$, of invoked methods. Such a conversion from scores to values may be derived by (a) empirically determining the distribution of correct score values by analyzing for a large data set for each method and then (b) deriving a value p from a *score* and the method-specifically given score-distribution. The advantage of operating with probability values is that they are **comparable and combinable** with each other.

A simple approximation of probabilities is to convert scores into "probabilities" according to a linear distribution function, i.e. to linearly shift score values with $-1 \leq score \leq +1$ to probability approximations $0 \leq p \leq +1$ by setting $p = \frac{score+1}{2}$. Experience has shown that treating this approximations exactly as if they were probabilities still leads, particularly for the finally incorporated methods, to sufficiently accurate and desirable results. The rest of the section therefore refers to probabilities\ only and leaves the choice for the particular conversion step implementation open.

For the final approach praxis s-values and t-values are defined to be *logodds*[58] and therefore directly computable from probabilities, logodds were chosen for technical reasons (see later in this section) and inherit the desirable comparability and combinability properties from probabilities.

[57] SDH (Schusterreiter-Dickson-Hofacker) contains the initials of the persons that contributed to the algorithm. Barry Dickson, the project supervisor, suggested to treat identity assignment as an optimization problem and was a driving force during development. Ivo Hofacker suggested the use of dynamic programming and provided a first algorithm prototype in a mathematical context. Christian Schusterreiter reformulated the problem statement and the algorithm such that it matches the given problem and embedded the result algorithm into the postprocessor module, where it solves occlusions.

[58] The logodds are also known as log-odds or *logit* values a corresponding probability.

Before invoking the algorithm a video is split into alternating σ and τ sequences and s-scores resp. t-scores for at least one s-method and t-method are computed. The chosen methods incorporated into the system are a size-based method and a point-based method, in particular $siz1$ and $boc6$ (see section 5.2.2.1).

Method $siz1$ uses a statistical test, particularly a variant of the sign test that provides good approximations for short sample sizes, in order compare fly size attributes $AreaEC$; the method computes the probability.for one fly being bigger resp. smaller than the other.

For each unoccluded sequence two flies are detected, these flies are arbitrarily named fly A and fly B. For each such unoccluded sequence the probability p that A is the bigger fly is computed to method $siz1$. By exploiting the fact that Drosophila males are smaller than females method $siz1$ may solve occlusions for standard male/female courtship assays.

From such a probability p the logodds s may easily derived, logodds are given by taking the logarithm of the odds and odds are given as the probability divided by its counter-probability, therefore $s = \ln(\frac{p}{1-p})$. Such odds resp. logodds values have desirable mathematical properties. Odds are intuitively much like chances, e.g. 50:50, multiple odds may therefore be multiplied with each other. Thus, by simple logarithm rules, multiple logodds may be added to each other.

Such logodds s-values are then assigned to each unoccluded sequence (see cyan values in figure 45(a)), positive values indicate that fly A is the bigger fly, negative values that it is assumed that B is the bigger fly.

Then, 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, in case of the method $boc6$ by tracking the fly perimeters through the occlusion (see 5.2.2.1 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 45(a)).

With s-values and t-values modeled like in figure 45 the occlusion resolvment problem may now be treated as an optimization problem, the SDH-Algorithm uses the **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 non-occluded sequences between them. But, most importantly it does *not* affect the identities in sequences *outside* these two occlusions, all sequences in Φ before τ_i resp. after τ_j remain unchanged.

Figure 45(b) depicts the flow of identifiers in (a) 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, in the second occlusion identities are swapped back, making "flip" a *local* operation only.

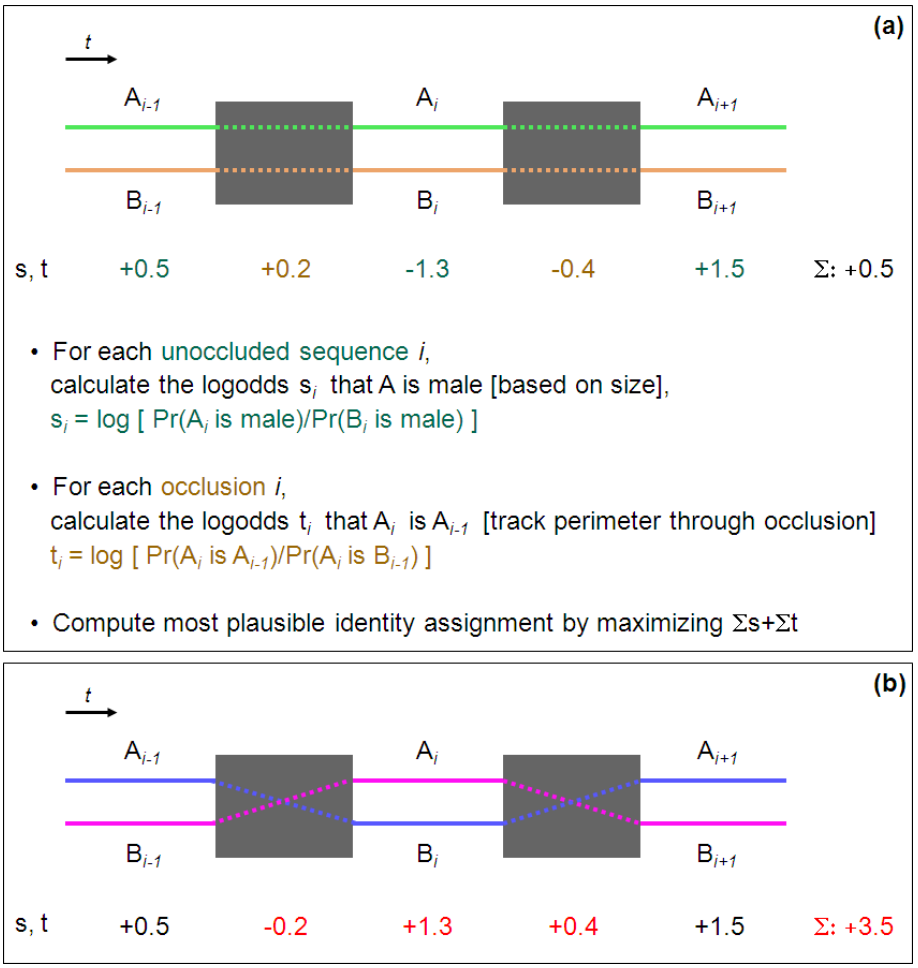


Figure 45: Sample problem instance for the SDH-algorithm (a) identified flies over time: A in green and B in orange. Occluded sequences marked by gray boxes. s-values associated with non-occluded sequences in blue, t-values for occluded sequences in brown. The text summarizes the computation of s-values and t-values and color-codes keywords and definition in blue and brown for s- resp. 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.

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

Lemma 50 *Let s be the s-value for a sequence σ, $s = \ln(\frac{p}{1-p})$. The s-value s' for the same sequence σ' with identifiers flipped is $-s$, $s' = -s$.*

The proof follows from the fact that a probability $p\prime$ for a flipped sequence σ' is to $1 - p$ and from simple logarithm arithmetic.
Proof. $s' = (\ln(\frac{p}{1-p}))' = \ln(\frac{p'}{1-p'}) = \ln(\frac{1-p}{p}) = \ln(\frac{1}{\frac{p}{1-p}}) = -\ln(\frac{p}{1-p}) = -s$ ■

Conclusion 51 *$t' = -t$, for t being the t-value for a sequence τ, $t = \ln(\frac{p}{1-p})$ and t' being the t-value for the same sequence τ' with swapped identifiers.*

Example 52 *Let the probability p that fly A is bigger than fly B be $p = 0.3$, and its corresponding logodd s-value therefore be $s = -0.8473$. Ergo the probability that fly B is bigger than fly A is $p\prime = 0.7$ and its logodds $s\prime = 0.8473$.*

Example 53 *Let the probability that fly A before the occlusion is fly A after the occlusion be $p = 0.8$ and its corresponding logodds t-value therefore be $t = 1.3863$. Ergo the probability that fly A before the occlusion is fly B after the occlusion is $p\prime = 0.2$ and its logodds $t\prime = -1.3863$.*

Example 54 *Let a t-score, i.e. the returned certainty value of method boc, be 0, its linearly shifted probability value p therefore be $p = 0.5$ and its corresponding logodd t-value therefore be $t = 0$. Ergo the probability of the swapped identities $p\prime$ and its logodds $t\prime$ are $p\prime = 0.5$ and $t\prime = 0$.*

In the figure 45 the flipped identities in (b), having a total value $\sum s + \sum t = 3.5$ are more plausible than the previous identities with $\sum s + \sum t = 0.5$.

Formally a flip operation is defined as follows:

Definition 55 *Let Φ be a partitioning of V' and $\mathcal{S}$ and $\mathcal{T}$ 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-values for τ_i. The operation $flip(i, j)$ on $\mathcal{S}$ and $\mathcal{T}$, defined as function $(\mathcal{S}, \mathcal{T})' = flip(i, j, \mathcal{S}, \mathcal{T})$, reverts the signs of t_i and t_j and of all $s_k, i \leq k < j$ in between them.*

Conclusion 56 *Let v be the values of $\mathcal{V}$, a merged sequence of alternating s-values from $\mathcal{S}$ and t-values from $\mathcal{T}$, starting with t-value t_1. Further let $\mathcal{O}$ and $\mathcal{E}$ define the set of odd resp. even indices for $\mathcal{V}$, $\mathcal{O} \subset \mathcal{V}, \mathcal{E} \subseteq \mathcal{V}$, such that all $i \in O$ refers to a value v_i coming from $\mathcal{S}$, $v_i = s_{\frac{i}{2}}$ and all $k \in \mathcal{E}$ refer to a value v_k coming from $\mathcal{T}$, $v_k = t_{\frac{k+1}{2}}$. The function $flip(i, j)$ on $\mathcal{V}$, defined as function $\mathcal{V}' = flip(i, j, \mathcal{V})$, requires that i and j are odd, $i, j \in O$, and is defined as*
$v_i = -v_i$
$v_j = -v_j$
$\forall k \in \mathcal{E}, i < k < j : v_k = -v_k$

Conclusion 57 *Another equivalent definition for* $flip(i,j)$ *on* $\mathcal{V}$ *is* $\forall x \in \mathcal{X}_{i,j}, \mathcal{X}_{i,j} = \{v_i, v_j | i,j \in O\} \cup \{v_k | k \in \mathcal{E}, i < k < j\} : v_x = -v_x$

Example 58 *Let* $\sigma_{i-1} \preceq \tau_i \preceq \sigma_i \preceq \tau_j \preceq \sigma_j$ *be a subset of a partitioning* Φ *of frames* $V\prime$ *and* $s_{i-1} \preceq t_i \preceq s_i \preceq t_j \preceq s_j$ *the associated* s *and* t *values. After the operation* $flip(i,j)$ *the altered sequence would be* $\sigma_{i-1} \preceq \tau'_i \preceq \sigma'_i \preceq \tau'_j \preceq \sigma_j$ *with values* $s_{i-1} \preceq t'_i \preceq s'_i \preceq t'_j \preceq s_j$ *resp.* $s_{i-1} \preceq -t_i \preceq -s_i \preceq -t_j \preceq s_j$. *This example corresponds to the one illustrated in figure 45.*

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

Notation 59 *Let* $flip(i,j)$ *and* $flip(k,l)$ *be flip operations on* V. *The combined operation involving both flips is denoted in set-from 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 also concatenable $flip(i,k) \cup flip(k,j) = flip(i,j)$ since $flip(k,k) \cup flip(k,k) = \varnothing$.

Lemma 60 *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 encourage the definition of a normal form $\mathcal{F}_\perp$.

Definition 61 *Let* f *be a flip operation* $f = flip(i,j)$, $f \in \mathcal{F}$, $|f|$ *denote the number of sequences affected by flip operation* $flip(i,j)$, $|f| = |\mathcal{X}_{i,j}|$ *and* $|\mathcal{F}|$ *be defined as* $|\mathcal{F}| = \sum_{f \in \mathcal{F}} |f|$. *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 then 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,* $\forall \mathcal{F}_i, \mathcal{F}_\perp \in \mathfrak{F} : |\mathcal{F}_\perp| \leq |\mathcal{F}_i|$.

Conclusion 62 *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* $flip(i,j) \in \mathcal{F}_\perp$.

Notation 63 *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* $\mathcal{F}_\perp = \{i,j,k,l\}$.

The properties of the flip operations allow to transform every set of flip operations $\mathcal{F}$ into its normal form $\mathcal{F}_\perp$ by elimination of double-flips and flip-overlaps that result in double-flipped sequences.

As stated above, the SDH-Algorithm aims to compute the most plausible identity assignment throughout the entire video by maximizing $\sum s + \sum t$ under the flip-operation, using the dynamic programming approach. Intuitively this allows s-methods and t-methods to complement and correct each other, especially in cases where an s-method is very sure but a t-method is not or vice versa.

When searching through all possible flip-operations it is sufficient to traverse normal forms $\{\mathcal{F}_\perp^i\}$ only, which reduces the infinite search space to an exponential search space. By sorting the non-overlapping flip-operations in ascending order intermediate results for all flip-operations up to a sequence τ_k may be reused, therefore the dynamic programming approach traverses the exponential search space within *linear time* and derives 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. The algorithms 64, 65, 66 and 67 provide a formal characterization of the SDH algorithm.

Figure 46 summarizes the workflow for the occlusion resolving strategy so far. At first a partitioning Φ is computed that splits the video frames into non-occluded sequences σ and occluded sequences τ. For each non-occluded sequence all attributes are computed and the size-based method $siz1$ is invoked such that a logodd s-value may be computed and assigned to each σ sequence. For each occluded sequence the keypoint-based method $boc6$ is invoked and a logodd t-value is computed and assigned to each τ sequence. While s-values correspond to the probability that fly A is the smaller fly and thus the male fly, 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.

Finally a dynamic programming approach, the SDH Algorithm (see algorithm 67 below) is used to maximize $\sum s + \sum t$ under the flip-operation and thus to assign the most plausible identity assignment throughout the entire video.

Algorithm 64 *function* $S, T = DP\_initialize(s, t)$

$n = |s|$
$T_{1,-1} = -\infty$
$T_{1,+1} = 0$
for $i \in \{1..n\}$
  for $c \in \{-1, +1\}$
    $S_{i,c} = T_{i,c} + c * s_i$ /* *add score if sign positive, subtract if negative* */
  end
for $c \in \{-1, +1\}$

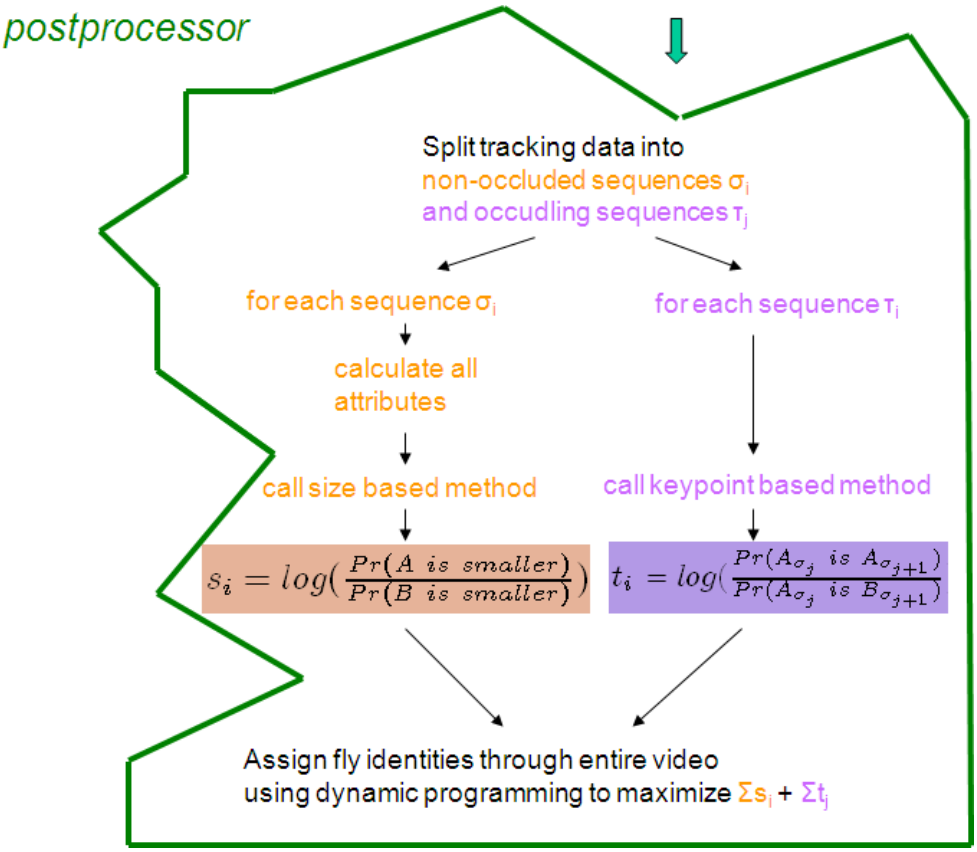


Figure 46: Workflow for resolving occlusions.

$T_{i+1,c} = max(S_{i,c} + t_{i+1}, S_{i,-c} - t_{i+1})$ /* *either add to same score or subtract from flipped score - take whatever is higher* */
end
end
end-function

Algorithm 65 *function* $flippos = backtrack(S, T, s, t)$
$c = 1$
for $i = n + 1$ *down to* 2
if $T_{i,c} = S_{i-1,c} + t_i$
$flippos_i = 0$
else
$flippos_i = 1$
$c = -c$
end
end

Algorithm 66 *function* $s\prime, t\prime = bulkflip(flippos, s, t)$
$fp_i = -(flippos_i \cdot 2 - 1)$ /* fp *is* -1 *if flippos was true,* $+1$ *otherwise* */
$t'_i = t_i \cdot fp_i$ /* $t := -t$ *at flippositions, remains the same otherwise* */
$s'_i = s_i \cdot \prod_{j=1}^{i} fp_j$ /* $s := -s$ *between flippositions, remains same otherwise* */
end

Algorithm 67 *function* $s\prime, t\prime = SDH(s, t)$
$S, T = DP\_initialize(s, t)$
$flippos = backtrack(S, T, s, t)$
$s\prime, t\prime = bulkflip(flippos, s, t)$
end

In particular, the SDH-algorithm listed as algorithm 67 consists of the following steps

1. A Dynamic Programming initialization step (see algorithm 64), where the properties of the video sequences resp. its s-values and t-values are traversed once in order to compute the cumulative scores S and T, where $S_{i,c}$ 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 resp. $c = 0$ 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 first occluded sequence is initialized with $T_{1,-1} = -\infty$ and $T_{1,+1} = 0$, this enforces the first (probably artificial) occlusion to remain unchanged. Note that the total cost of the identity assignment is given in $T_{n+1,1}$.

2. A backtracking step (see algorithm 65), 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 66), where the result flip operations in $\mathcal{F}_\perp$ are applied to the initially given s-value and t-values in order to derive the flipped scores s' and t' of the optimal solution, $\sum s' + \sum t' = T_{n+1,1}$.

Definition 68 *Whenever a s-value or a t-value is v is 0, it is replaced by $v = \epsilon$, where ϵ is the smallest representable floating point number that can carry a sign. least value carrying a sign.*

This replacement is done for purely technical reasons, a typical value for ϵ is $\epsilon = 2.2251 \cdot 10^{-308}$. Since ϵ holds a sufficiently small number it does not affect the result of the SDH-algorithm, however, it provides the nice technical property that flip operations applied to such this value $s' \in \{-\epsilon, \epsilon\}$ are tracked.

For each sequence σ_i a value $swap_i = \frac{s'_i}{s_i}$ with $swap_i \in \{-1, +1\}$ may be computed, in case $swap_i = -1$ the identifiers for sequence σ_i have to be swapped. Since $\min_{i \in \mathcal{S}} |s_i| \geq \epsilon$ the division is guaranteed to be defined. This final step is used to apply the result of the SDH-algorithm to fly identifiers in all sequences in Φ resp. in all frames in V.

Combining probabilities for multiple methods of same type

Methods of same type may further be combined to come up with an improved score, the tree-combination in section 5.2.2.1 highlighted one approach for combining t-methods. The probability-converted score of Independent methods may further be combined by the *Dempster-combination [18][19]*, which allows to mathematically combine evidences from different sources into a combined degree of belief. The Dempster combination is defined as follows:

Definition 69 *Let e_1 and e_2 be two independent evidences. The Dempster combination of these two evidences is defined as $e_1 \otimes e_2 = \frac{e_1 \cdot e_2}{1-K}, K = (1 - e_1) \cdot e_2 + e_1 \cdot (1 - e_2)$.*

Conclusion 70 *The Dempster-combination $\otimes$ forms a commutative group with identity element $e = 0.5$.*

Definition 71 *Let $\otimes$ be the Dempster-combination. The Dempster combination $\bigotimes$ for a set of evidences $\{e_i\}$ is defined as $\bigotimes\{e_i\} = e_1 \otimes \bigotimes\{e_i\} \backslash e_1$, with $\bigotimes \varnothing = 0.5$.*

The definition above allows to combine evidences resp. probability-converted scores from an arbitrary number of methods, the empty set of evidences results in a combined probability of 0.5, which results in logodds-values of 0 resp. ϵ and corresponds pure guessing.

Note that this definition further allows to invoke the algorithm with a s-method only, where all t-evidences are set to $\varnothing$ and all t-values therefore set to 0 resp. ϵ, or with a t-method only, with all s-values set to ϵ. The combined maximum $\sum s + \sum t$ of such degraded cases delivers the same assignment that the single method would derive. The strategy comparison framework may therefore, by selectively choosing s and t method incorporation, compare the performance of individual and combined methods.

Performance and Efficiency of the SDH-algorithm and Conclusion of Approach II

In a test set of 11 male-female courtship videos with a manually annotated ground truth the algorithm assigned 1041 out of 1048 occlusions correctly and thus had 99.2 % accuracy. The 7 wrong assignments affected 20 out of ~150000 unoccluded frames and thus has a frame-wise accuracy of 99.99 %. The ~150000 correctly assigned frames correspond to about 1 hour 45 minutes of the recorded video, the 20 wrong frames are all together less than a second.

These results are derived since the algorithm implementation implicitly minimizes the number of non-occluded frames. This property is inherited from the approximation of the sign test that is currently used as s-method, the probability it computes is dependent on the size difference *and* on the length of the sequence the s-value is computed for. For this reason weak scores typically come for short non-occluded σ sequences, which then may be dominated by its surrounding τ sequences.

Further, due to the properties of the algorithm, it "self-corrects" its own mistakes. Potential errors typically occur *pairwise*, one real error immediately followed by a second one that compensates the first one, 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 they do not affect the rest of the video.

These two error patterns, pairwise errors ensuring *local* misassignments only and *short* non-occluded sequences as potential error domains, explain the low number of non-occluded frames that are misidentified, a very desirable property of the algorithm.

Another essential property of the algorithm is that it runs very efficiently. It runs in linear time $\mathcal{O}(|\Phi|)$, applying the optimal solution and deriving initial s and t values also runs in linear time $\mathcal{O}(|V|)$. The algorithm once passes the sequences resp. their s and t values from beginning to the end, and then from the end back to the beginning. Given an arbitrary set of such sequences, after

these two passes the algorithm knows, out of all thinkable flip-combinations, the smallest set of flip operations that is required to transform the arbitrary initialization into the optimal solution. Knowing the required flip-operations it can apply them to deliver the optimal identity assignment.

Experiences where ground-truth annotating users downgraded the automatically pre-annotated ground-truth's quality suggest that the automatic occlusion resolvment method may in many cases be more reliable than a human annotator. However, the automatically derived occlusion assignments may still be manually overruled with the annotationTool (see section 6.2 p. 164).

5.2.3 Other Methods

This section contains short summaries for a set of useful methods for checking the validity of a frame (see section 5.2.3.1), for handling manual annotations (5.2.3.2) and for resolving heading (see section 5.2.3.3).

5.2.3.1 Checking Frame Validity

A given set of primary attributes may be reported as suspicious in case the values are not conform to a given norm. Experience has shown that the eccentricity corrected area attribute *AreaEC* (see figure 43 on p. 107), defined as ,$AreaEC = Area\sqrt{1 - EccentricityC}$[59] has stable values and is particularly useful for this purpose, frames with *AreaEC* outside the mean +/- three standard deviations, $AreaEC \tilde{} \mathcal{N}(\mu, \sigma^2), |AreaEC - \mu| > 3\sigma$, are marked as suspicious and their primary attributes are rather interpolated (see section 5.2.4) than used.

The most common reason for suspicious values are missegmentations, the method above allows to automatically detect such misdetected frames.

Technically suspicious values resp. missegmentations are treated like occlusions and successive invalid frames correspondingly as longer occluded sequences. Therefore an identity assignment for the flies before the missegmentation to the flies after the missegmentation is computed and all primary attributes are interpolated. Missegmentations occur very rarely and missegmentation sequences typically consist of just one or extremely rarely two frames and the occlusion resolving and interpolation methods work particularly well for such short sequences.

5.2.3.2 Handling Manual Annotations

The annotationTool (see section 6.2) allows a user to manually overrule machine decisions.

The upper picture in figure 47 depicts a correct occlusion assignment, the green line marks the current position, the data bar shows the *isOcclusion* attribute with the current occlusion in magenta, the *Area* attribute of the two flies and the red and the blue line the identity assignment from the last frame before the occlusion to the first frame after the occlusion. The lower picture

[59] *EccentricityC* contains the *Eccentricity* values where high values are smoothed by the logistic function, $EccentricityC = \frac{Eccentricity}{1+e^{5*Eccentricity}}$

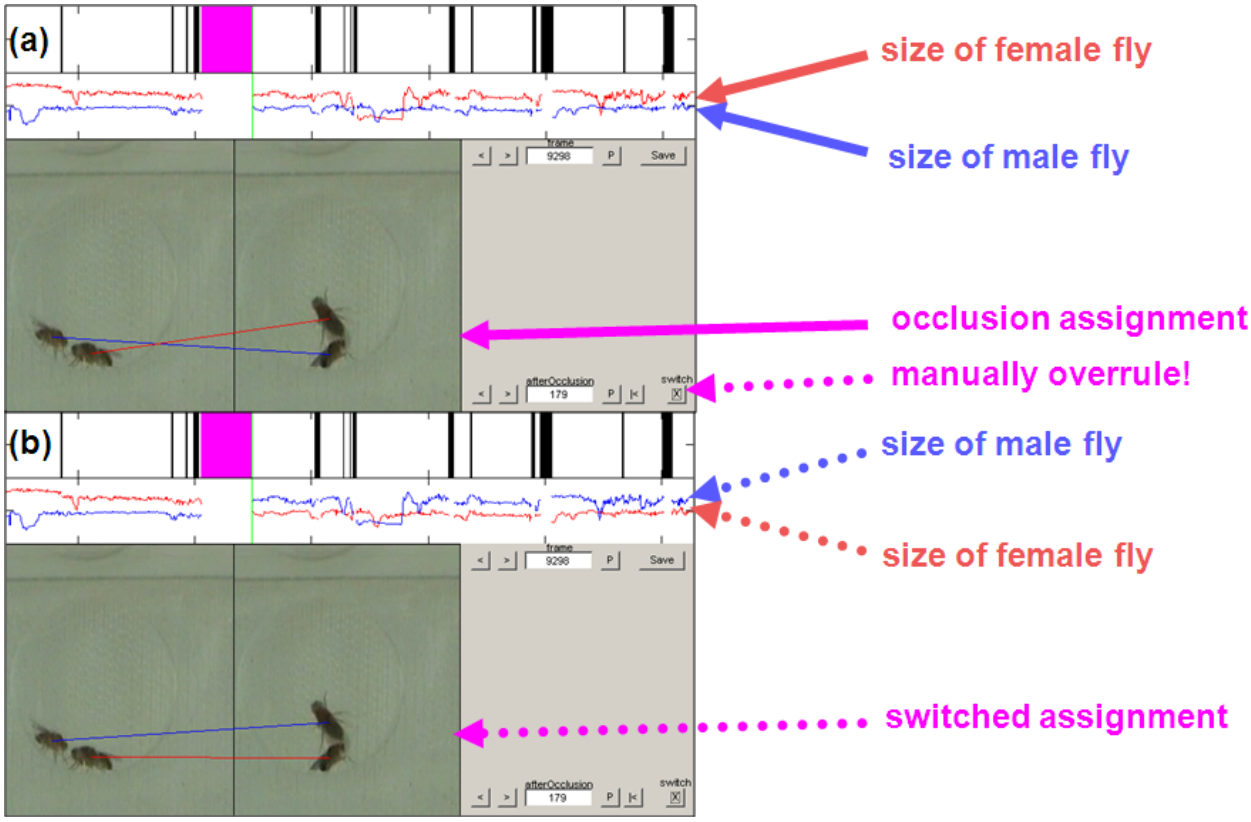


Figure 47: Manual overrulement of an occlusion solution. The picture above depicts a correct occlusion assignment, the picture below a manually overruled (but then wrong) identity assignment.

depicts the same scene after manually overruling the former correct machine decision. The red and the blue line now wrongly point to the other flies and the values of the *Area* attribute are swapped from the position of the green line on up to the end of the video.

When manually reviewing occlusion assignments the annotationTool allows to sort occlusions by a confidence score and thus to inspect the most difficult occlusions until the reviewer is convinced that the remaining occlusions are correct as well.

However, in case a reviewer overrules the machine in a difficult case the postprocessor module has to incorporate these annotations after resolving occlusions but before further downstream computation. As long as the tracked

and the manually annotated time series are, except for the annotations, identical incorporating manual annotations is a straightforward task.

The machine operates with a time series t^-, which resembles the tracked time series after identifier assignment in sigma sequences (see section 5.2.1.2) and a mapping m, which is the machine derived permutation to map the identifiers in t into a $\{1, 2\}$ domain (see sections 5.2.1.3 and 5.2.2.2). The application of the permutation m on t^- results in a time series t with machine assigned identifiers. Note that t^- unambiguously determines an occlusion map o that marks then beginning and end for each occlusion in t^- resp. t.

The aim of handling manual annotations is to incorporate a manual information for a time series a into a new machine track t although it is not guaranteed that a and t were computed with the same tracker version.

The annotated time series a also determines an occlusion map $o\prime$ marking beginning and end for occlusions in a. As long as the two occlusion maps o and $o\prime$ match the information from the annotated time series a may be incorporated by computing the identifier permutation $m\prime$ that would map t into a and then apply $m\prime$ to m before it is used to set the identifiers to a value in $\{1, 2\}$.

However, in case the body regions in one of the two time series are tracked differently this may affect the observed occlusions sequences. In order to exploit the given manual annotation the machine will take over annotations for all unambiguous cases.

Therefore, for every frame, the centroids two flies in a are matched to the centroids of the two flies in t by the Hungarian algorithm introduced in section 5.2.1.2 p. 80. Whenever a matching is found, the identifiers of the manual annotation are taken over, otherwise the computed machine identifiers are used as default values.

This step allows to compute a identifier computation $m\prime$ that exploits the manual annotation in the best possible way. Since the quality of the tracker output typically comes with an improved body segmentation, where the fly bodies were separated in frames where they were detected as occluded before, the typical application pattern is a case where a new time series t contains sequences $\tau_i, \sigma_i, \tau_{i+1}$ where the old annotated time series a just detected a long τ_j sequence that starts the first frame in τ_i and ends with the last frame in τ_{i+1} For this particular case the manual annotation for σ_{i-1} to σ_{i+1} may still be incorporated but for the newly separated unoccluded sequence σ_i, which was not inspectable for the user when annotating a, the default machine scores are applied.

Such newly occurring σ sequences are typically very short, however, they are still marked to be reviewed for the next time when a user inspects the time series.

5.2.3.3 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 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. And finally, the head does typically not flip by 180 degrees within a single frame.

Interestingly the heading problem may be reduced to the occlusion problem described in section 5.2.2, and the proposed SDH-algorithm of section 5.2.2.2 may be applied to solve heading as well.

The idea is to model every single frame as a σ sequence and "artificial gaps" between frames as τ sequences. The evidence from movement and wings are incorporated as s-methods, note that s-methods have to operate on attributes that are comparable through the entire video, and the known persistency constraint is incorporated as a t-method, such t-methods are applied for every τ sequence. When computing logodd s-values that correspond to the probability that point A is the Head of the fly and logodd t-values that correspond to the probability that point A in the frame before τ is point A in the frame after τ.

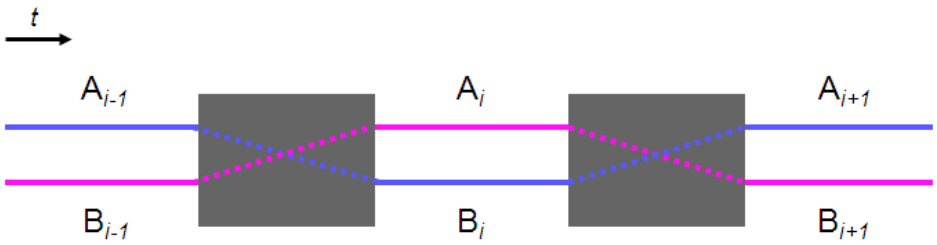
Figure 5.2.3.3 depicts how to model the heading problem as the same optimization problem as the occlusion problem. Note the strong similarities between figure 5.2.3.3 and figure 45 on p. 113, the text written in red in figure 5.2.3.3 marks the few differences.

In order to resolve heading it is sufficient to first define the s- and t-methods that incorporate movement, wing anatomy and persistency 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 of the $Centroid$ C and $wCentroid$ W are determined for every frame.

Definition 72 *Let X_i denote the value of point X in frame i and XY denote the euclidean distance between point X and Y. The s-score $score_{move}$ is defined as $score_{move} = \frac{A_iC_{i-1} - B_iC_{i-1}}{A_iC_{i-1} + B_iC_{i-1}}$.*

Definition 73 *The s-score $score_{wing}$ is defined as $score_{wing} = \frac{AW - BW}{AW + BW}$*



Same problem as occlusions!

- 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 $\Sigma s+\Sigma t$

Definition 74 *The combined s-score* $score_{move \otimes wing} = \max(score_{move}, score_{wing})$.

The score $score_{move}$ will be positive whenever the fly moved rather in A than in B direction, and 0 if it moves sideways, the score $score_{wing}$ will be positive in case A is closer to the centroid of the wing region. Note that $-1 \leq score_{move} \leq +1$ and $-1 \leq score_{wing} \leq +1$ and that the two scores are combined by a simple maximum aggregation rather than the dempster combination introduced in 5.2.2.2. From the combined score $score_{move \otimes wing}$ probability approximations and finally logodd values s may be derived as in the occlusion case (see section 5.2.2.2).

Definition 75 *The t-score* $score_{persist}$ *is defined as* $score_{persist} = \frac{AB+BA-AA-BB}{AB+BA+AA+BB} Eccentricity$.

The persistency score $score_{persist}$ is $-Eccentricity \leq score_{move} \leq +Eccentricity$, rescaling the score by the *Eccentricity* attribute ensures low persistency scores when flies "turn upwards" and are thus round, from such a position they are may abruptly change their heading via z-direction. The logodd t-value derived from that score is further weighted by a empirically determined factor of 6 that makes flipping of $Head$ and $Tail$ less attractive.

The SDH-algorithm (see algorithm 67 defined on p. 118) will compute the most plausible heading assignment for all video frames by maximizing $\sum s + \sum t$ under the flip-operation introduced and discussed in the occlusion section 5.2.2.2.

For the heading case the linear time characteristic of SDH-algorithm is essential, heading is typically computed for 15000+ frames, and quadratic algorithms already become unhandy for such large problem instances.

5.2.4 Interpolation Across Occlusions

Before resolving occlusions the postprocessor splits occluded sequences into two *possible worlds*, which allows compute attribute sets for each of the two possible occlusion resolving assignments. The postprocessor does not yet concern about resolving occlusions, but pre-computes interpolated trajectories and primary attributes for each of the two possible resolutions of the occlusion. When occlusions are resolved, the world corresponding to the occlusion assignment may be chosen while the other one is discarded. Figure 48 (left) below shows the intuitive workflow, where identities are assigned and the data is then interpolated accordingly; on the right the figure depicts the workflow involving multiple possible worlds, which allows to pre-compute interpolation *before* solving occlusions.

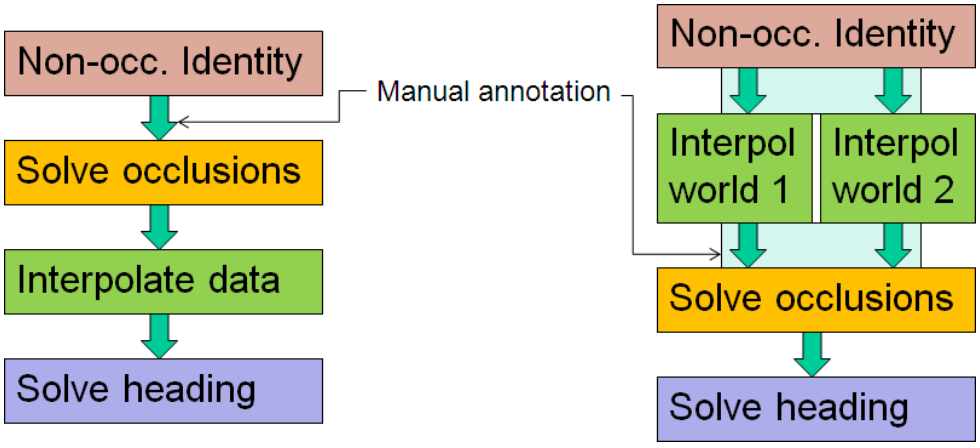


Figure 48: (left): linear workflow where identities are assigned and data is interpolated accordingly. (right): data is interpolated first and split into two possible worlds. Later, when occlusions are resolved, the worlds are merged and the data flow continues linearly.

The purpose this precomputing step is to in enable occlusion strategies that incorporate the coverage of the observed occluded blob by estimated interpolated ellipses, and thus prefer the more feasible interpolation when resolving occlusions. Further, the precomputing step allows to compute the interpolation step much earlier, and prepares to shift interpolation into the tracker module, which would make sense for performance reasons.

Note that it is sufficient to provide interpolation for primary attributes only, since all further attributes may be re-calculated from those primary attributes. The paragraphs below describe how to derive interpolated body area attributes, the same steps may also be applied to wing area attributes accordingly in case

the *wing assignment during occlusions* (see last paragraph of this section) is not applicable.

5.2.4.1 Interpolating Ellipse Characteristics

The *Centroid* coordinates of fitted ellipses are interpolated by applying a cubic spline interpolation [6] to the x and y coordinates of the centroid. The interpolation of a tau sequence τ_i is fit from values taken from the flanking sigma sequences σ_i and σ_{i+1}. Within a window of l frames information of up to n sigma frames in both directions is incorporated, default values for these parameters are $l = 20$ and $n = 5$. In case the length of sequence σ_k contains less frames than required, $|\sigma_k| < n$, then $n - |\sigma_k|$ frames of neighboring sequence σ_{k-1} resp. σ_{k+1} are incorporated until a total of n frames are found or the window of length l does not contain further frames.

In case the cubic spline interpolation would result in coordinate values outside of an arena, the more conservative cubic pchip interpolation [6] is used instead. Figure 49 below shows interpolation results of the two methods for two different scenarios, the blue markers show the known values, the green line resp. the dotted red line the interpolation results of the two methods. The figure depicts overshooting effects of the spline method for the sine-like trajectory in subfigure (a) and overshooting effects for the plateau-like trajectory in subfigure (b). It further depicts that the cubic pchip interpolation has less oscillation and guarantees no overshoots.

The pchip method produces feasible values inside an arena since all incorporated data points are guaranteed to have inside arena values, the arenas are circular and pchip guarantees no overshoots. However, spline interpolation are known to produce smoother results [6][60], and especially the often observed circular trajectories along the arena borders are fit extremely well.

The $MinorAxisLength$ and $MajorAxisLength$ attributes of the fitted ellipses are subjected to a simple linear interpolation across the occlusion. Based on these two values the primary attributes *Area* and *Eccentricity* may be derived by ellipse formulas with $a = \frac{MajorAxisLength}{2}, b = \frac{MinorAxisLength}{2}$, $Area = a \cdot b \cdot \pi$ and $Eccentricity = \sqrt{1 - \left(\frac{b}{a}\right)^2}$. Further, due to the intrinsic convexity of ellipses, $ConvexArea = Area$.

The *Orientation* angle is also subjected to a simple linear interpolation, but may progress in either rotation directions, clockwise or counter-clockwise. The problem is easiest described by showing an interpolation where ellipses are rotated in the *wrong* direction, as depicted in figure 50. The left picture shows

[60] c.f. [6] section pchip for more details and a comparison of the two methods

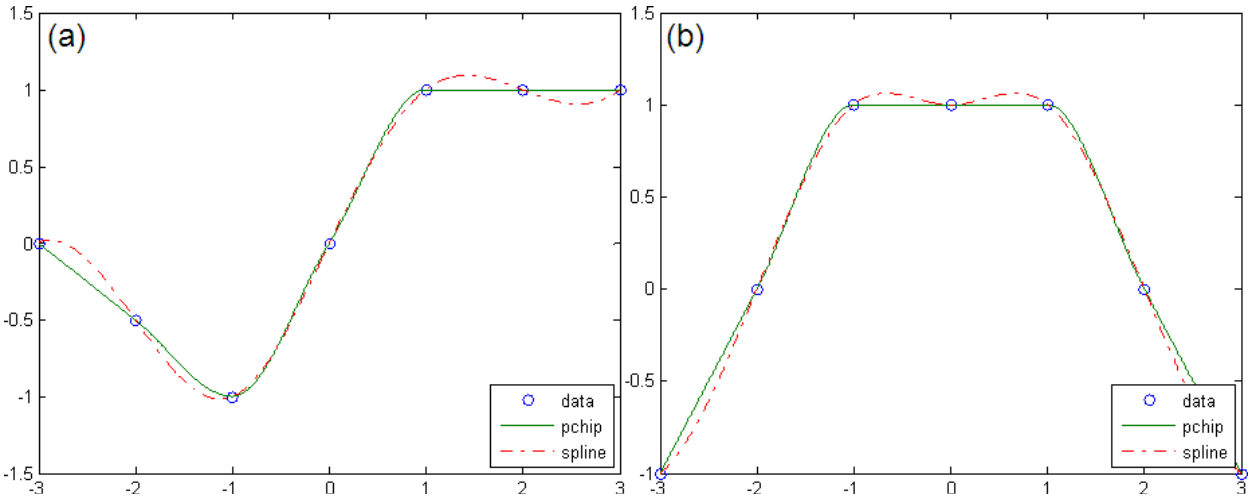


Figure 49: Two sample interpolation for different methods (a) sine-like scenario (b) plateau-like scenario. Blue rings depict the original data points, the green line the pchip interpolation, the red dotted line the spline interpolation.

the scene before an occlusion, the picture to the very right the scene after the occlusion, for all frames in between ellipse characteristics are interpolated. Note that the difference in orientation of the female flies (in red) before and after the occlusion is almost 90 degrees, and both rotation directions feasible. However, it is very obvious that the red ellipse in the middle picture does *not* capture the corresponding fly body very well.

In order to improve interpolation results rotation directions are therefore estimated by the following rules:

First, the angle differences $\Delta Orientation$ are computed for both rotation directions, and directions are named *short* and *long* according to their lower resp. higher angle difference value. In case $\Delta Orientation_{short}$ is less than 60°, the short rotation is chosen.

Otherwise the rotation direction is determined by measuring how well the interpolated ellipse would cover the merged body areas in to be interpolated sequence. For this reason, for each interpolated frame, both short and long rotations are computed, and an overlap measure is computed as follows:

Definition 76 *Let E be the region containing a filled interpolated ellipse for one fly, B be merged region containing the two fly bodies and $|X|$ denote the number of pixels in region X. The overlap measure o is then defined as $\frac{|E \cap B|}{|E|}$.*

Definition 77 *Let τ be an occluded sequence and oi be the overlap measures for each frame of that sequence. The overlap measure for the whole sequence is then defined as the mean of overlap measures, $\mathfrak{o}^\tau = mean(o_i)$.*

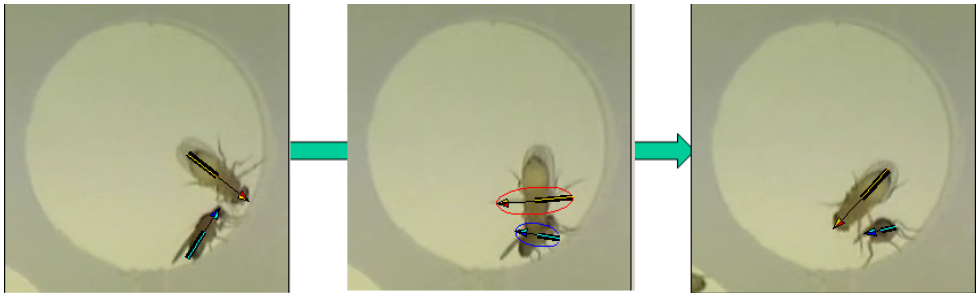


Figure 50: Wrong ellipse rotation interpolation. The left and the right picture show the frame before resp. after the occlusion. The picture in the middle one of the interpolated pictures. The interpolated ellipse in the middle picture is obviously rotated in the wrong direction.

In case the difference of the two average overlap measures is greater than 0.6, $\mathfrak{o}^{\tau}_{short} - \mathfrak{o}^{\tau}_{long} > 0.6$, the direction with the *short* direction is chosen; otherwise the ellipses are interpolated in *long* direction.

direction, and that the method maximizes the total overlap produced by ellipse rotations for both flies, (short,short), (short,long), (long,short) or (long,long), such that they best fit the observed merged body region.

Figure 51 shows a sequence of interpolated ellipses, drawn over the original merged body region. While the rotation selection criteria works well for the more often observed short occluded sequences and short rotations, the figure depicts a (randomly chosen) long sequence with a $(short, long)$ rotation, i.e. a short ellipse rotation for the male fly and a long rotation for the female fly. Since rotation velocity is not constant during the observed sequence, the linear interpolated values are a bit off in the middle. However, the interpolated values provide a feasible approximation the fly bodies during the occluded sequence.

Having interpolated all five primary attributes characterizing the interpolated ellipses, the last primary attribute *BoundingBox* may be derived. Further, the Orientation values for alternative rotation, named *alternativeOrientation*, and an *alternativeBoundingBox* are computed kept for debugging purposes.

5.2.4.2 Wing Extensions During Occlusions

A final interpolation step aims to use the interpolated ellipse information to split the merged wing region in a feasible way, such that wing extension events (see **??**) may be derived *during* occlusions.

The algorithm below often allows a reliable detection of such wing extension events, these events commonly occur during very short, slightly touching

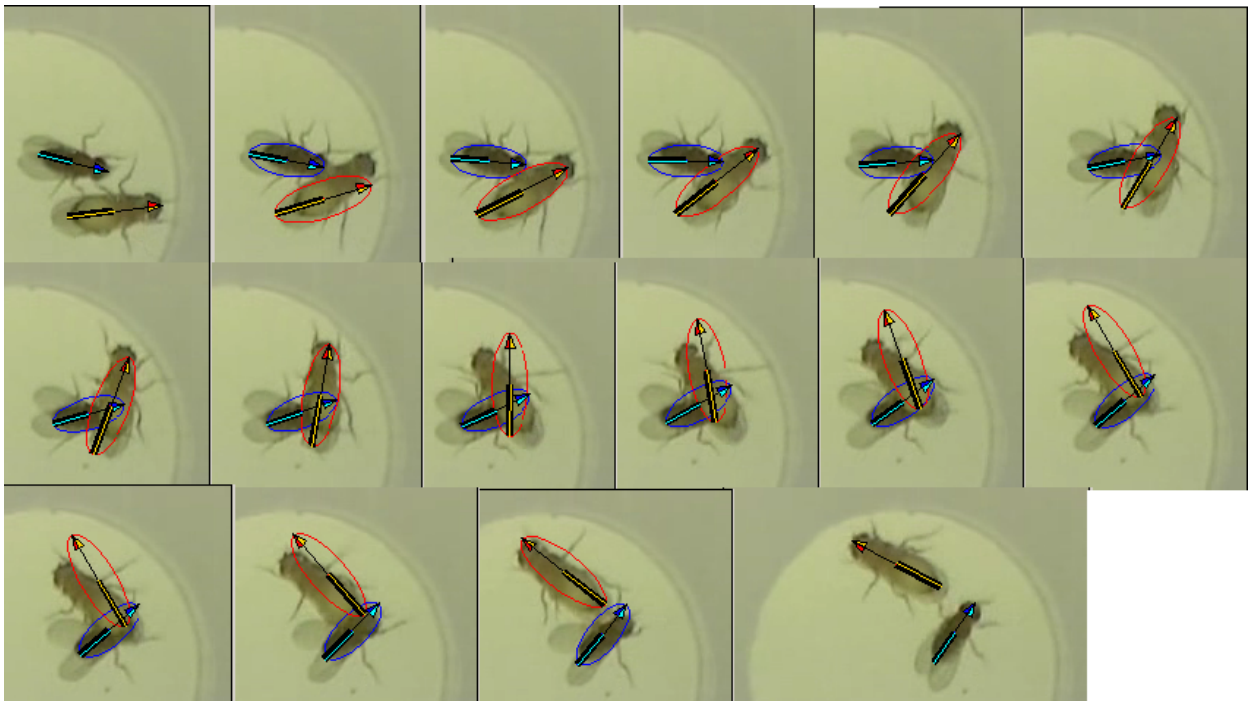


Figure 51: Sequence of interpolated ellipses. The first and the last picture show frames before resp. after the occlusion. The frames in between depict the sequence of interpolated ellipses, overlayed on the original occluded video frame. The female fly (red) rotates in long, the male fly (blue) in short direction.

occlusions, where the available interpolated ellipses are expected to be well approximated. Figure 52 depicts such a scene, two slightly merged fly bodies \(left) are well-approximated by two ellipses (right), such that it is feasible to capture the ongoing singing event. The merged wing region is therefore split (very similar to section 4.2.3) and the resulting wing areas are assigned to the two flies.

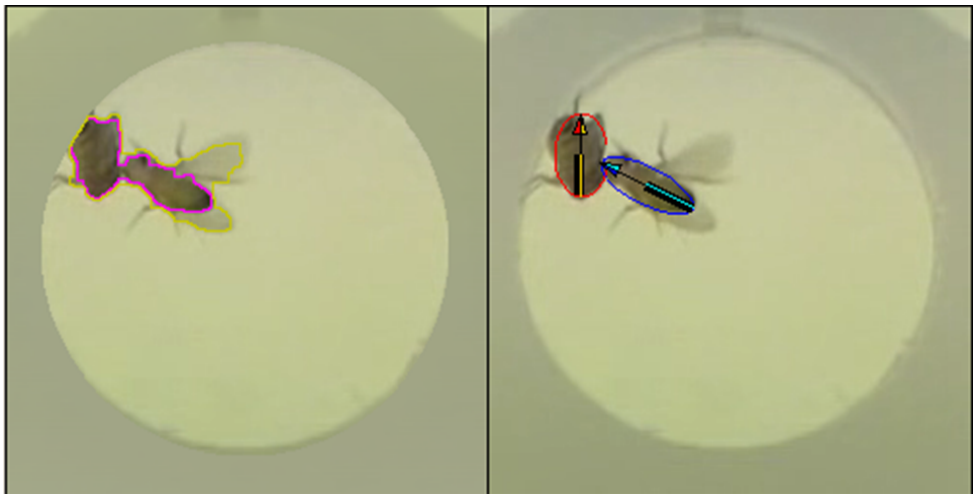


Figure 52: Rescued Wing extension during occlusion event. (left): the body regions of the two flies are slightly occluded. (right): the interpolated ellipses fit to the flies body regions and allow to still capture the ongoing Wing extension event.

In particular, the merged wing area is split between the two flies by two consecutive floodfill separation[61]. The first floodfill separation is applied to the merged body region B, using the two interpolated ellipses as the two concurrent seeds, thus assigning each merged body region pixel to an ellipse, resp. a fly, and thus splitting the merged body region into two new body regions B_1 and B_2, $B_1 \cup B_2 = B$. The second floodfill separation is then applied to the merged wing area W, using the two newly separated body regions B_1 and B_2 as seeds. The derived wing areas W_1 and W_2 may then be used to derived primary wing attributes per fly for each interpolated frame.

This procedure is not guaranteed to be valid, in particular not for long or heavily convoluted occlusions. In order to decide whether to trust the split wing areas, the overlap measure for the seeding ellipses or length and minimal body area of the occlusion may be considered. Note that in case an interpolated ellipse is way off and not even touching the merged body region, the split body region split will give the trivial result $B_A = B$ and $B_B = \varnothing$, such that degraded

[61] see section 4.2.3, p. 69

cases may easily be detected and the primary wing attributes are then set to *missing*. Note further, that the wing area during occlusions is by default not assignable and that wing attributes are *missing* during occlusions by default; the method above allows to overcome this shortcoming in many cases and to detect events still correctly. It may still happen that it splits a wing region in an undesired way, however, the filtering mechanism for event detection (see section 5.2.5.2, p. 144) further takes care to keep falsely detected events rare.

The *wing extension during occlusion* method rescues the detection of the singing event in figure 52 and in many similar scenes.

5.2.5 Event Detection

So far the focus of the document was to describe different data cleaning, transformation and normalization steps from raw video data into a structured time series with containing various attributes. The goal of all these processing steps is to enable the analysis of specific patterns within videos. The courtship behavior and its underlying subbehaviours constitute such patterns or *events* within the time series resp. the video.

The aim of the event detection is to find courtship sequences within the video material.

When geneticists manually analyze videos they protocol every time they see a specific behavior which is aimed to be observed within the video, e.g. Drosophila courtship behavior. The number and duration of observed behavior events is counted resp. *scored* in a video, and aggregation of these scores, e.g. the courtship index, which is the percentage of time the flies court, are used to summarize the behaviour shown in that video.

When searching for answers how courtship behaviour is encoded in genes, e.g. by a genom-wide screen, people record thousands of courtship videos and then assess the behavior they see. This is a time-consuming and tedious task, and it produces subjective scores. Further the outcome has low information content as complex behavior is assessed by a single number.

The overall goal of the system, as stated in section 1.3, was to come up with an automated and thus high-throughput and objective method that can measure the courtship index and further to capture the complex behavior in all its details. Within the proposed solution, the courtship index resp. its subbehaviours are automatically derived by a video processing step that recognizes and tracks flies, followed by a statistical classification step that finds courtship sequences. The question covered within this section is how a behavioural pattern resp. an *event*, in this particular case a sequence of courtship behavior, is characterized by observable variables and found within the video material resp. the time series data. Given the steps so far, the event detection is the last missing stone of the puzzle.

Detecting courtship events and discriminating its sub-behaviours turned out to be a challenge itself, especially definition and selection of attributes with biologically relevant information content. The approach here was to first define as many potentially useful attributes as possible, essentially *everything* that appeared to make sense was computed for a set of test videos, from which some final characterizing attributes were chosen. Since event detection aims to characterize behavioural patterns out of observable variables, a large starting set

The event detection problem was attacked by two opposed approaches, the bottom-up approach (see section 5.2.5.1) where machine learning techniques are applied in order to learn event characterizations from user scores, and the top-down approach (see section 5.2.5.2) where events are classified according to user-specified rules.

The next sections describe the two approaches how events are detected and classified, the system supports application of *both* approaches.

5.2.5.1 Bottum-up Classification

The bottom-up approach was the first implemented event detection method and aimed to derive a courtship index according to a user-annotated training-set that specified the behaviour. When the project was started there was no formal description characterizing courtship behaviour available, however, biologists were able to assess courtship behaviour when it was observed. One of the first aims of the project was to turn this implicit knowledge into an **explicit characterization**, such that a robust assessment of behaviour and data comparison between different manual scorers is possible.

The main principle of the bottom-up approach is that the system *learns* which video sequences to classify as an event based on manual scores given by a human scorer. In particular the *logistic regression* approach is used for machine learning.

Logistic regression: Computing a Courtship Index

The logistic regression [20][21] is a supervised machine learning approach that learns how to describe states based on observable variables. It then allows binomial regression, i.e. to predict a discrete outcome, such as a group membership, from a set of variables. Logistic Regression comes without any pre-assumptions about distribution or 'nature' of such a variable. The goal is to classify each frame either as a 'courtship' or a 'non-courtship'-frame.

Figure 53 summarizes the logistic regression approach in a nutshell. The method consists of two steps, a supervised learning step and a step where the learned parameters are applied. In the application step the probability that a frame is a courtship frame is computed out of observable (extracted) attributes which are weighted by learned parameters. Those parameters are learned by a maximum likelihood estimation that maximizes the plausibility of the parameters on a given data set, this done by an iterative procedure where different solutions are tried until the smallest possible deviation between observed and

predicted data, the *best fit*, is found. Therefore the maximum likely parameter for each attribute is stepwise computed, and then the most significant attribute–parameter-pair is incorporated. (see figure 53 (1))

More particular, the logistic regression is a generalized linear model that uses the logistic curve to relate learned attribute-parameter pairs $\beta_i w_i$ to a probability p.

When fitting the model, it is assumed that the logits[62]

$logit(p) = \ln(\frac{p}{1-p}) = \sum \beta_i w_i$

may be linearly related to a combination of observable variables weighted by parameters, and the aim of the fitting process is to derive the maximum likelihood attribute-parameter pairs, it therefore selects the attribute which most improves the model fit and determines its parameter weight β_i.

Given the model parameters $\{\beta_i\}$ the probability of an event, in this case of courtship group membership, may be derived from observable variables as

$p = \frac{1}{1-e^z}, z = \sum \beta_i w_i$

The observed attributes are weighted by learned parameters and the resulting linear combination is used to parameterize the logistic function $\frac{1}{1-e^z}$.

The logistic regression approach was implemented using the Matlab functions $glmfit$ and $glmval$, early experiments about its general applicability were done in SPSS.

Application and Results

As a first step for every supervised machine learning approach a training set is required, in this particular case the training set consisted of 10 video chambers showing different courtship behaviour. These video chambers were manually scored by five different biologists, each aiming to annotate courtship behaviour.

The very first attempt was mainly intended to check whether machine learning approaches are suitable, and in particular how well the annotated scores are reproduceable from simple observable variables. It was unclear whether the implicit definition of courtship followed complex descriptions or simple characteristics and rules. In order to gain a first impression the five biologist were advised to score as if they would score for for their own purposes and only frames where the five scorers agreed in their classification as courtship or non-courtship were picked for the training and test set. The result was that by looking at simple distance and angle measures the machine was able to reproduce a score with 80 % accuracy. Further, when the flies were identified through the entire video

[62] Logits and logodds (from the occlusion section 5.2.2.2) are essentially the same, the logodds of an event are the logit of the probability for that event, however, for logistic regression the term logit is commonly used in the literature.

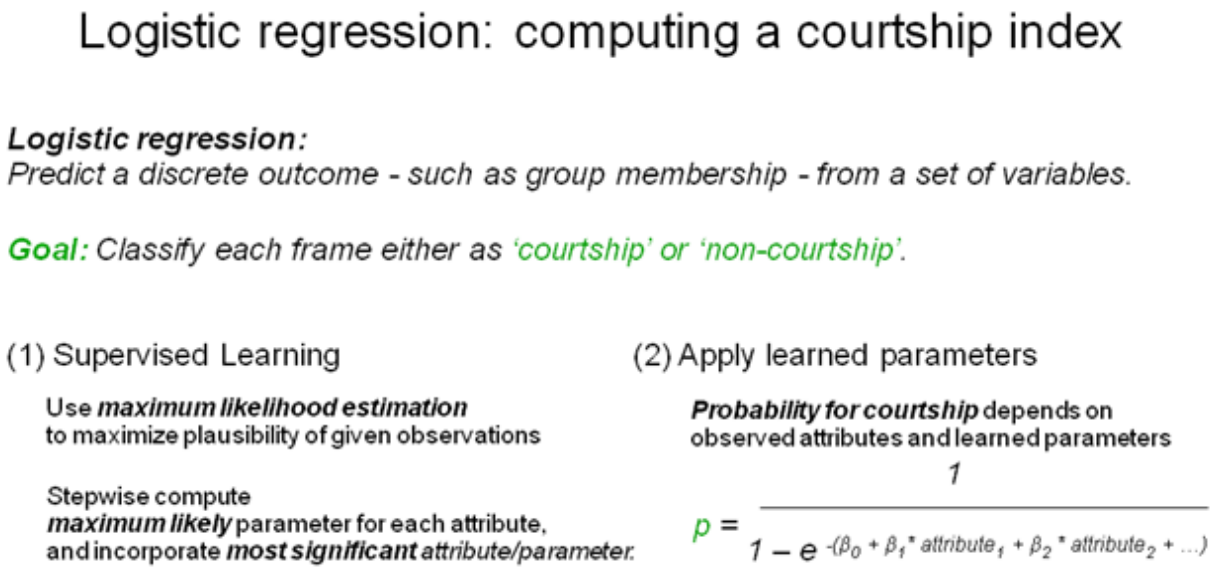


Figure 53: Logistic regression in a nutshell.

reproducibility was boosted to 90 %, and with further given heading information accuracy was 95 %.

Besides the logistic regression approach some alternatives, e.g. a classification and regression tree as introduced in section 5.2.2.1 were evaluated, however, it turned out that all tested approaches delivered very similar results in overall accuracy. The logistic regression approach was chosen rather arbitrarily because it comes as a single formula and because one can intuitively explore the parameter space. For example, given a set of observable variables and the resulting probability for courtship, single parameters may easily be varied and analyzed, e.g. if the fly was a millimeter further away or the orientation angle was altered by some degrees, how would the result probability be affected?

The very first goal was to come up with a logistic regression classifier that is not the outlier when comparing it to the five manually annotated scores. However, when analyzing the distribution of the manual scores (figure 54 (a) gives an impression) it turned out that they strongly disagreed. People seemed to score consistently with themselves but there were apparently different schools of scoring.

For this reason the videos were re-annotated, this time not by everyday-scoring annotation but by most exact scoring possible. This meant that people were able to pause the video, to watch difficult passages repeatedly and in slow-motion, and in particular to *edit* their scores. This extensively annotated data was more suitable for machine learning approaches than the real-time recordings of the very first attempt.

Figure 54 (a) shows a comparison of the extensively annotated data (plotted in x-direction) versus the real-time annotated data of five scorers, the scores are aggregated as a courtship index which aggregates the percentage of time that the male fly courts. The blue data points are particularly interesting as they depict the very same scorer, once in real-time and once scoring extensively. The data points of the black real-time scorer are almost identical with the extensively annotated scores, however, the red and the magenta scorers differ significantly, mainly due to a different school of scoring.

The extensively annotated data was then fed into the logistic regression algorithm introduced above in order to choose attributes with high information content that are suitable for reproducing the biologists annotations.

The "winning" attributes for computing the courtship index are introduced in the following itemization and in figure 55. They are

- $distHeadTail$: specifies the distance between one flies head to the other flies tail (see figure 55 (a)), normalized by the arena radius r, i.e. $\frac{distanceHeadTail[pixels]}{r[pixels]}$.

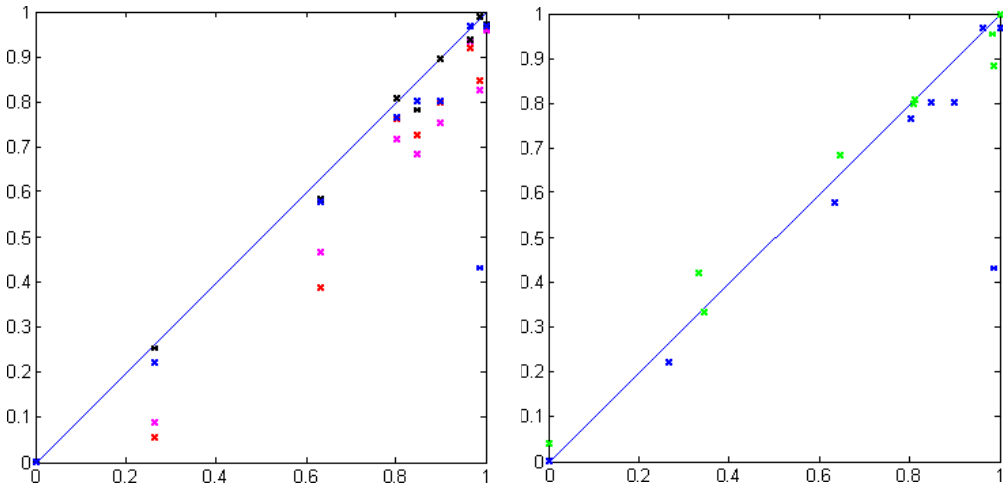


Figure 54: (a) everyday-scores of 5 biologists plotted against a ground truth-score of one biologist. (b) everyday-score of the ground truth scoring biologist and the machine learned scores plotted against the ground truth score

The *Head* and *Tail* are defined as the two points where a line in angle of *Orientation* that crosses the *Centroid* leaves the body region (for definition of *Centroid* and *Orientation* see section 4.2.4, section 5.2.3.3 describes how to discriminate *Head* from *Tail*).

- *nArea* and *nwArea*: specifies the normalized *Area* of the body region (*nArea*) and the wing region (*nwArea*), normalized by r^2. (see figure 55 (b), the blue region, the cyan+blue region and the red region are the incorporated values).
- *flyLen*: MajorAxisLength of the body region (see figure 55 (c))
- *Eccentricity*: the *Eccentricity* attribute (see section 4.2.4) for the body region (see figure 55 (d))
- *minDist*: the minimum of the distances between *Head* to *Head*, *Head* to *Tail*, *Tail* to *Head*, *Tail* to *Tail* and *Centroid* to *Centroid*, each referring to the body region (see figure 55 (e))
- *distance*: the distance between the two fly body centers resp. *Centroids* (see figure 55 (f))
- *hrnDirectednessR*5: a combined distance and angle measure that turned out to be particularly useful.

Definition 78 *Let P be the intersection point of the two straight lines unambiguously defined by the fly Centroids and Orientations. The attribute directedness id defined as the distance between P and the other flies Centroid.*

Flies *orienting towards the other* fly therefore have lower directedness values than flies that don't, and flies *closer* to the other fly also have lower values, low values are indicators for potential courtship activities. (see figure 55 (g))

Definition 79 *Let $bHeadedness$ be a Boolean attribute that is 0 when a fly orients towards the other fly and 1 when it orients away from it. The attribute rnDirectedness is defined as the reciprocal value of the directedness value normalized by the arena radius, $nDirectedness = \frac{directedness}{r}$, $rndirectedness = \frac{1}{nDirectedness}$, and $hrnDirectedness$ is a heading guided value and defined to be 0 when the fly orients away from the other fly, $hrnDirectedness = rndirectedness \cdot headedness$.*

Large values of hrnDirectedness indicate potential courtship activities. Taking the square root of *rndirectedness* turned out to increase the specificity of the attribute, taking higher order roots further improved the information content for the logistic regression purposes.

Definition 80 *$hrnDirectednessR5$ is defined as $hrnDirectednessR5 = \sqrt[5]{rndirectedness} \cdot headedness$.*

- *$nAngleHeadHead$*: the angle between the following points: $Tail$ of the fly, $Head$ of the fly and $Head$ of the other fly (see figure 55 (h)), normalized by taking its cosine. Angles towards the other flies head therefore result in positive normalized values with a maximum 1 when oriented directly at the other flies head, angles away from the other fly result in negative values with a minimum of -1 when orienting away resp. when orienting the tail directly to the other flies head and or values of 0 when the fly orients sideways.

The order of the attributes above resp. in the table below resembles the order in which they were chosen by the learning algorithm.The resulting weights β_i, computed by the learning step and usable to compute an automatic courtship classifier out of observable variables are:

β_i	w_i	fly
−2.8378	1	
−5.7163	$distHeadTail$	1
59.2156	$nwArea$	1
−160.6717	$nArea$	1
−73.9275	$nArea.2$	2
10.7874	$flyLen.2$	2
10.4516	$Eccentricity$	1
−3.5416	$minDist$	1
2.28061	$distance$	
0.43	$hrnDirectednessR5$	1
0.0587	$nAngleHeadHead$	1

The table above shows the parameters parameters β_i and observable attributes w_i that are incorporated for courtship classification, in particular the probability for a frame being a courtship frame is defined as $p = \frac{1}{1-e^z}, z = \sum \beta_i w_i$. The third column of the table indicates which's flies attributes are incorporated, an empty entry indicates that the attribute involves both flies. The first row specifies the constant value β_1.

Figure 54 (b) depicts result scores versus the extensively annotated data. The real-time scores of the very same scorers are again plotted in blue, the scores automatically derived by logistic regression learning and application are depicted in green. The results of the first attempts suggest that the automatic classifier does as well as the human scorers whose data was used for video analysis so far. However, the automatic classifiers provides a robust and reproducible way for scoring and allows high throughput courtship assessment.

The software allows to merge a time series with other user-defined time-series, this allows to feed in manual annotations of various behaviours and to import further data points into the system, like e.g. audio recordings of a courtship song. This enables to operate with extended attribute sets and to learn new bottom-up classifiers. Whatever behaviour the manual annotations refer to, the system will aim to derive a classifier based on a set of suitable observable attributes.

5.2.5.2 Top-down Classification

Besides the data-driven bottom up-classifier that provides a courtship index (see 5.2.5.1 above) the system further contains *top-down classifiers* for the *individual courtship steps*.

The top-down approach allows a biologist to specify classification rules that define behaviours. Such classification rules define a binary classification for a

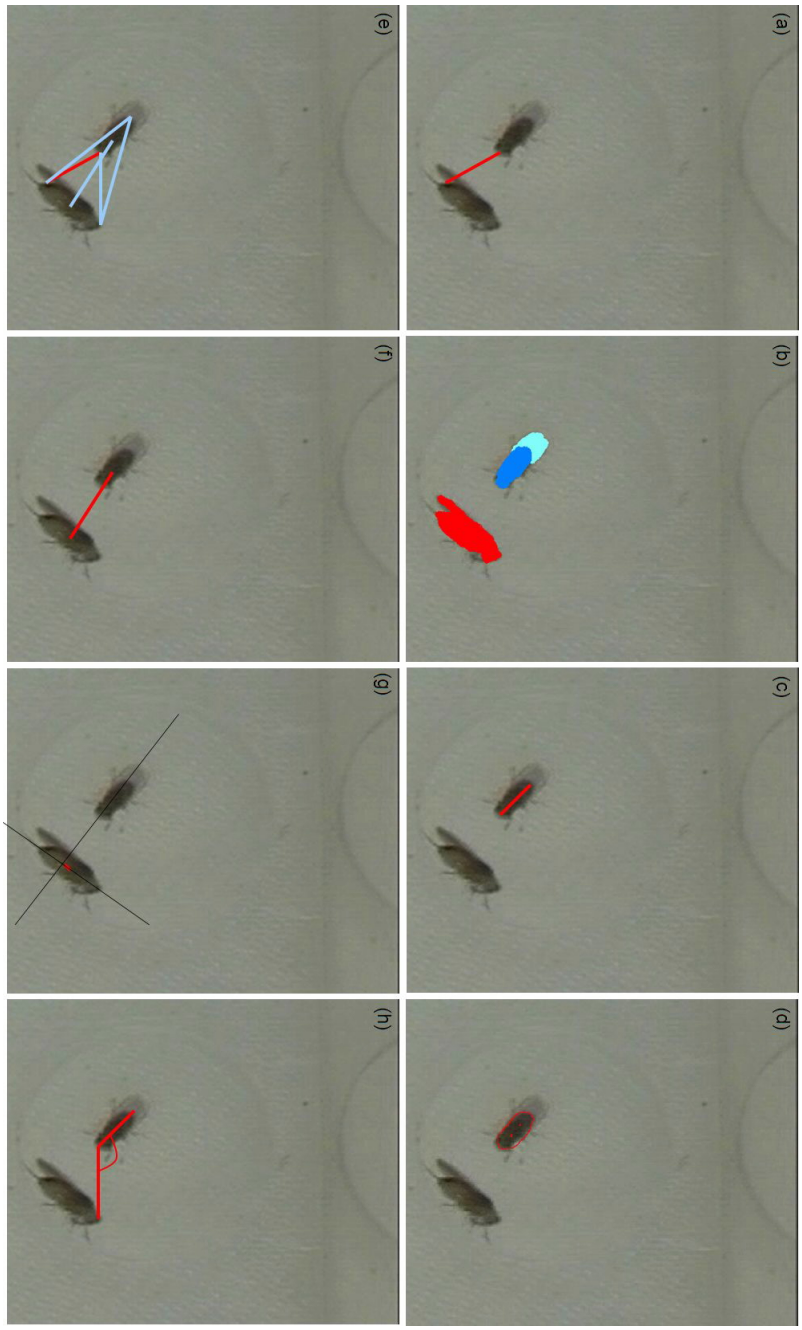


Figure 55: Winning attributes in pictures: (a) $distHeadTail$, the distance between the flies $Head$ to the other flies $Tail$ (b): $nArea$ and $nwArea$, the size of the body resp. the wing area (c) $flyLen$, the $MajorAxislength$ of the fitting ellipse and the distance between $Head$ and $Tail$ (d) $Eccentricity$, a measure for *roundness* that depends on the two focal points of the covering ellipse resp. on the relation between its $MajorAxisLength$ and $MinorAxisLength$ (e) $minDist$ the shortest of the five depicted distances (f) *distance*, the distance between the two flies centers-of-gravity (g) $hrndirectednessR5$, a combined measure of orientation angle and distance (h) $nAngleHeadHead$, the angle between a fly and the other flies $Head$.

specific behaviour based on observable primary or downstream computed attributes. In biology expert definitions are a standard approach for behaviour specifications.

Drosophila courtship is a very rich behaviour that comes in several steps (see section 1.1). The following section introduces expert definitions for *following*, *orientation*, *copulation*, *circling* and *wing extension* sub-behaviors.

Further it is detected when the two flies are close together such that the fly bodies melt to a single region, which detects a *touch* behavior resp. *legTouch* behaviour if the two regions are separated by a very thin connection that is potentially a fly leg.

Each top-down classifier definition consists of a number of conditions that must hold for the classifier to fire. These conditions are Boolean sub-parts of the classifier, condition attributes are named by a behaviour abbreviation and a letter, e.g. FollB for condition "B" for the *following* detector. All condition values are filtered by a moving median filter, the window length for all condition filter steps is 5 [frames].

The classifier itself is true when all if its sub-conditions are true and for each classifier a *filtered* classifier is defined that is computed by a morphological opening of the classifier values. This step filters out short events and guarantees a minimum duration for the detected events. The opening operation takes the required continuity of the event [in seconds] as a parameter, this continuity parameter is individually defined for each classifier.

Here the definitions for the top-down classifiers and their underlying attributes.

The first attribute used is *movedG*, which is short for Gaussian movement, it is an alternative to the framewise movement attribute *moved* and incorporates the movements between time t_1 and t_2.

Definition 81 *Let t be frame for which movedG is computed, $Centroid_t$ denote the Centroid of a fly at frame t and $Centroid^x$ and $Centroid^y$ denote the x resp. y coordinate of a Centroid. Further let $t_1 = t - 2$ and $t_2 = t + 1$, and let $\widehat{Centroid}_{t_1}$ be the mean of $Centroid_{t_1}$ and $Centroid_{t_1+1}$ and $\widehat{Centroid}_{t_2}$ be the mean of $Centroid_{t_2}$ and $Centroid_{t_2+1}$. The attributes deltaXG and deltaYG are defined as $deltaXG = \frac{\widehat{Centroid}_{t_1}{}^x - \widehat{Centroid}_{t_2}{}^x}{t_2 - t_1}$ and $deltaYG = \frac{\widehat{Centroid}_{t_1}{}^y - \widehat{Centroid}_{t_2}{}^y}{t_2 - t_1}$, and the attribute movedG is defined as $movedG = \sqrt{sqrt(deltaXG^2 + deltaYG^2)}$.*

Notation 82 *For each per-fly attribute a, a_x denotes the attribute value for fly x. When time and fly subscripts are used, the value for fly x attribute a at time t is $a_{t,x}$*

Definition 83 *Condition FollB is defined as $FollB = movedG_1 \geq Foll\_moved_{min} \wedge movedG_2 \geq Foll\_moved_{min}$ with $Foll\_moved_{min} = 2$ [mm/sec].*

Condition $FollB$ requires that both flies move.

Definition 84 *The attribute distance is defined as the euclidean distance between the two fly Centroids.*

Definition 85 *Condition FollC is defined as* $FollC = distance \geq Foll\_distance_{min} \wedge distance \leq Foll\_distance_{max}$ *with* $Foll\_distance_{min} = 2$ *[mm] and* $Foll\_distance_{max} = 5$ *[mm].*

Condition $FollC$ requires a specific distance between the two flies resp. it requires the two flies to be close, but not too close, together.

Definition 86 *Let midline be straight line that crosses a flies Centroid in angle Orientation. The attributes Head and Tail mark the two points where midline crosses the flies Perimeter.*

Definition 87 *For flies A and B the attribute* $angleHeadTCentroid_A$ *is defined as the angle between the points* $(Centroid_A, Head_A, Centroid_B)$

Definition 88 *Condition FollD is defined as* $FollD = angleHeadTCentroid \leq Foll\_angleheadcentroid_{max}$ *with* $Foll\_angleheadcentroid_{max} = 60$ *[°].*

Condition $FollD$ requires the following fly to orient towards the followed fly.

Definition 89 *The attribute* $moveAngleDiff$[63] *is defined as the difference between moveAngles of the two flies, the moveAngle is defined as the angle between the straight line defined by* $(Centroid_{t-1}, Centroid_t)$ *and the x axis.*

Definition 90 *Condition FollE is defined as* $FollE = moveAngleDiff \leq Foll\_moveangle_{max}$ *with* $Foll\_moveangle_{max} = 90$ *[°].*

Condition $FollE$ requires the two flies to move into the same direction.

Definition 91 *The attribute* $distHeadTail_A$ *is defined as the euclidean distance between* $Head_A$ *and* $Tail_B$.

Definition 92 *Condition FollF is defined as* $FollF_A = distHeadTail_A < distHeadTail_B$.

Condition $FollF$ requires the distance from the followers $Head$ to the followed flies $Tail$ to be smaller than the distance from the followed flies $Head$ to the followers $Tail$.

[63] The definition of this attribute is taken from [30], its alias name is also *perona_25*.

Definition 93 *The classifier Foll is defined as* $Foll = FollB \wedge FollC \wedge FollD \wedge FollE \wedge FollF$ *with* $Follcont = 1$ *[sec].*

Following events occur wherever the conditions $FollB$ to $FollF$ above hold for at least $Follcont = 1$ second. Figure 56 depicts a part of the time series that contains following events, it visualizes the $Foll$ classifier and all its underlying conditions and their underlying attributes.

Definition 94 *Condition OriA is defined as* $OriA = angleHeadTCentroid \leq Ori\_angleheadcentroid_{max}$ *with* $Ori\_angleheadcentroid_{max} = 30$ *[°].*

Condition $OriA$ requires the orienting fly to orient towards the other fly, the threshold of 30 [°] is much more strict that the threshold for condition $FollD$, which is 60 [°].

Definition 95 *Condition OriB is defined as* $OriB = distance \geq Ori\_distance_{min} \wedge distance \leq Ori\_distance_{max}$ *with* $Ori\_distance_{min} = 3$ *[mm] and* $Ori\_distance_{max} = 10$ *[mm].*

Condition $OriB$ requires a specific distance between the two flies, it is less strict than condition $FollC$, which requires a distance between 2 and 5 mm.

Definition 96 *Condition OriC is defined as* $OriC = movedG_1 \leq Ori\_moved_{max} \wedge movedG_2 \leq Ori\_moved_{max}$ *with* $Ori\_moved_{max} = 1$ *[mm/sec].*

Condition $OriC$ requires that neither of the flies move.

Definition 97 *The attribute distHeadHead is defined as the euclidean distance between the two flies Heads*

Definition 98 *Condition OriD is defined as* $OriD = distHeadHead < distHeadTail$.

Condition $OriD$ requires the orienting fly be closer to the other flies $Head$ than to its $Tail$.

Definition 99 *The classifier Ori is defined as* $Ori = OriA \wedge OriB \wedge OriC \wedge OriD$ *with* $Oricont = 1$ *[sec].*

Similar to following, orientation events occur wherever the conditions $OriA$ to $OriD$ hold for at least $Oricont = 1$ second. Figure 57 depicts the Ori classifier and all its underlying conditions and their underlying attributes for the same time series as in figure 56 above.

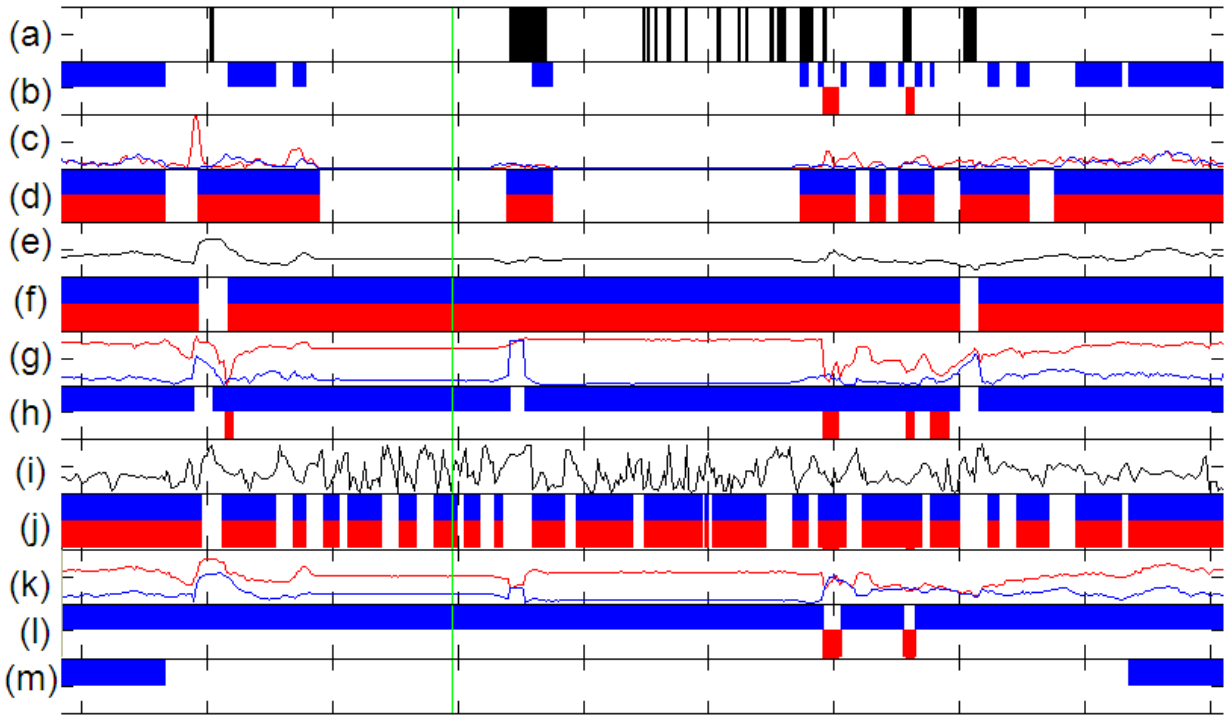


Figure 56: Visualization of following classifier and its underlying attributes. (a) *isOcclusion*, black indicates that the flies body regions overlap each other (b) *Foll*, binary classifiers are visualized by two half bars, the blue half bar is displayed where the attribute is true for the male fly, the red half bar is displayed when the attribute is true for the female fly (c) *movedG*, the blue line depicts values for the male fly, the red line for the female fly (d) *FollB* (e) *distance*, just one black line depicting the distance between the two flies (f) *FollC* (g) *angleHeadTCentroid* (h) *FollD* (i) *moveAngle* (j) *FollE* (k) *distHeadTail* (l) *FollF* (m) *Folled*, the continuity filtered classifier where events shorter than 1 second are eliminated. The screen shot in figure 60 depicts the pictures and all events for this sequence.(

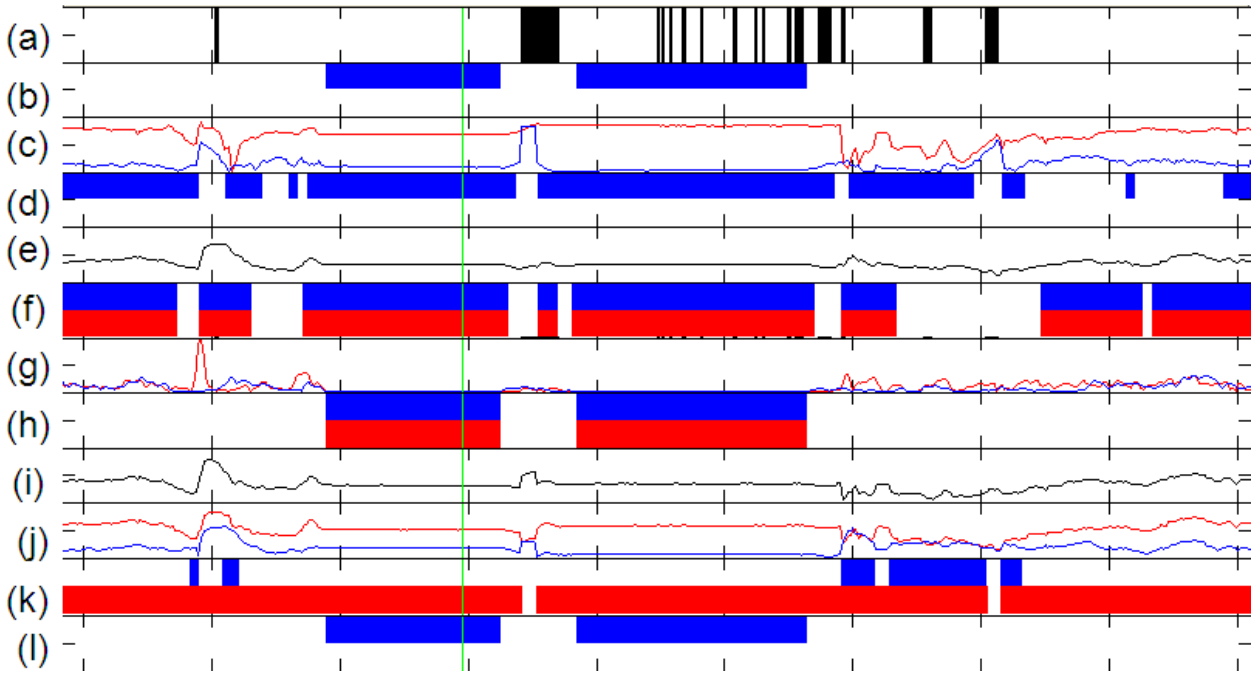


Figure 57: Visualization of orientation classifier and its underlying attributes, visualization format as in figure 56. (a) *isOcclusion* (b) *Ori* (c) *angleHeadTCentroid* (d) *OriA* (e) *distance* (f) *OriB* (g) *movedG* (h) *OriC* (i) *distHeadHead* (j) *distHeadTail* (k) *OriD* (l) *Oried*

Definition 100 *Condition CircA is defined equally to OriB.*

Definition 101 *Condition CircB is defined equally to FollD.*

Definition 102 *Condition CircC is defined as* $CircC_A = movedG_A \geq Circ\_moved_{max}$ *with* $Circ\_moved_{minThis} = 3$ *[mm/sec].*

Condition *OriC* requires that the circling fly moves at a minimal rate.

Definition 103 *Condition CircD is defined as* $CircD_A = movedG_B \leq Circ\_moved_{max}$ *with* $Circ\_moved_{maxThat} = 1$ *[mm/sec].*

Condition *OriD* requires that the target circled fly does not move too fast.

Definition 104 *The attribute* $angleDiffG$ *is defined as the angle difference between the Orientation and the moveAngle, where the moveAngle is defined as* $\arccos(\frac{deltaXG}{movedG})$*, with* $deltaXG$ *and* $movedG$ *as in definition* **??** *of movedG.*

Definition 105 *Condition CircE is defined as* $CircA = angleDiffG \leq Circ\_angleDiffG_{min}$

with $Circ\_angleDiffG_{min} = 30$ *[°].*

Condition *OriE* requires the circling fly to move sideways, at least in an angle of 30 degrees away from the flies *Orientation*.

Definition 106 *The attribute* vaz[64] *is defined as* $vaz = movedG \cdot \sin(angleDiffG)$.

Definition 107 *Condition CircF is defined as* $CircF = vaz \leq Circ\_vaz_{min}$

with $Circ\_vaz_{min} = 3$ *[mm/sec].*

Condition *OriF* requires the azimuthal velocity of the circling fly to be at least 3 mm/sec.

Definition 108 *The classifier Circ is defined as* $Circ = CircA \wedge CircB \wedge CircC \wedge CircD \wedge CircE \wedge CircF$ *with* $Circcont = 0.5$ *[sec].*

Circling events are defined to happen wherever the conditions *CircA* to *CircF* hold for at least $Circcont = 0.5$ seconds, figure 58 depicts the *Ori* classifier and all its underlying conditions and attributes.

[64] The definition of this condition was inspired by the supplementary methods of [30], where a similar attribute is used to define circling.

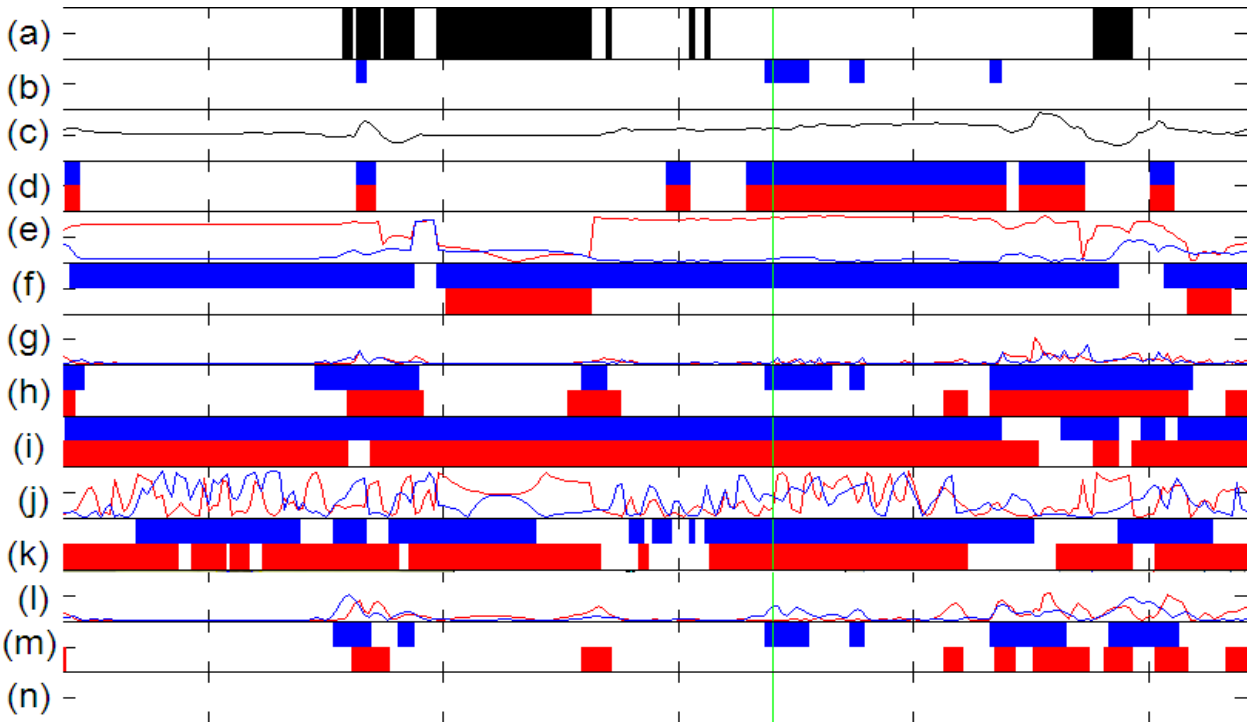


Figure 58: Visualization of circling classifier and its underlying attributes, visualization format as in figure 56. (a) *isOcclusion* (b) *Circ* (c) *distance* (d) *CircA* (e) *angleHeadTCentroid* (f) *CircB* (g) *movedG* (h) *CircC* (i) *CircD* (j) $angleDiffG$ (k) *CircE* (l) *vaz* (m) *CircF* (n) *Circed*

Definition 109 *Let $wRotPerimeter\_act$ contain the perimeter points of the wing region, $wRotPerimeter\_act \subset wPerimeter$, that are within an angle α, $10° < \alpha < 100°$, where α is the angle between midline and the straight line defined by a point $P \in wPerimeter$ and the $Centroid$. Further, let tWT be the point $P_\top \in wRotPerimeter\_act$ that is the furthest point from the $Centroid$, $|P_\top, Centroid| \gtrsim |P_\top, Centroid|$. The attribute $tWTangle$ is defined as the angle $\alpha_\top$, which is the angle between the midline and the straight line defined by tWT and the $Centroid$. The attribute $tWtangle^{left}$ is computed from the subset $wRotPerimeter\_act^{left} \subset wRotPerimeter\_act$ where all points $P^{left} \in wRotPerimeter\_act^{left}$ lie left to the flies midline directed from $Tail$ to $Head$, the attribute $tWTangle^{right}$ is computed accordingly from $wRotPerimeter\_act^{right}$.*

Definition 110 *The condition $WingAL$ is defined as $WingAL = tWTangle^{left} \gtrsim Wing\_minangle$, the condition $WingAR$ as $WingAR = tWTangle^{right} \gtrsim Wing\_minangle$*

with $minangle = 30$ [°].

The condition holds when the furthest point tWT of the wing perimeter segment $wRotPerimeter\_act$ is more than 30 degree from the *midline*, measured from the $Centroid$.

Definition 111 *Let $wRotPerimeter\_act$ be the same subset of $wPerimeter$ and let $tWarea$ the number of pixels of the wing area within the segment defined by $wRotPerimeter\_act$. The attribute $tWareaRatio$ is defined as $tWareaRatio = \frac{tWarea}{Area}$, and $tWareaRatio^{left}$ and $tWareaRatio^{right}$ are defined correspondingly upon the left resp. right subset of $wRotPerimeter\_act$.*

Definition 112 *The conditions $WingBL$ and $WingBR$ are defined as $WingBL = tWtareaRatio^{left} \gtrsim Wing\_minarea$ resp. $WingBR = tWtareaRatio^{right} \gtrsim Wing\_minarea$ with $Wing\_minarea = 0.3$.*

Condition $WingBL$ resp. $WingBR$ holds when the wing area segment surrounded by $wRotPerimeter\_act^{left}$ resp. $wRotPerimeter\_act^{right}$ has at least 0.3 times the flies body $Area$.

Definition 113 *The classifiers $WingL$ and $WingR$ are defined as $WingL = WingAL \wedge WingBL$ and $WingR = WingAR \wedge WingBR$.*

Definition 114 *The classifiers $Wing$ and $WingBi$ are defined as $Wing = WingL \vee WingR$ and $WingRi = WingL \wedge WingR$.*

Definition 115 *The attribute $flyLeft_A$ is true when fly B is left to fly A, seen from the directed midline pointing from $Head$ to $Tail$ from fly A.*

Definition 116 *The classifier $WingT$ is defined as $WingT = (WingL \wedge flyLeft) \vee (WingR \wedge \neg flyLeft)$, the classifier $WingA$ is defined as $WingA = Wing \wedge \neg WingT$.*

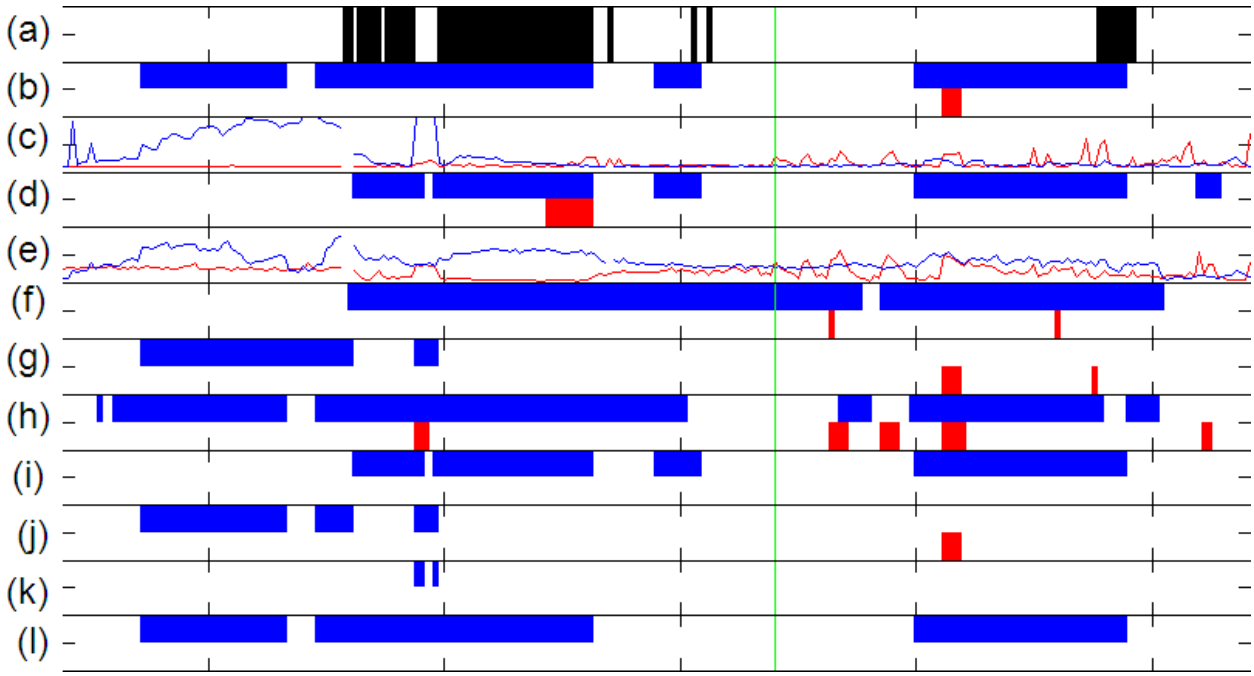


Figure 59: Visualization of wing extension classifier and its underlying attributes, visualization format as in figure 56. (a) *isOcclusion* (b) *Wing* (c) *tWTangle* (d) *WingAL* (e) *tWTareaRatio* (f) *WingBL* (g) *WingAR* (h) *WingBR* (i) *WingL* (j) *WingR* (k) *WingBi* (l) *Winged*

The classifiers above detect extension of the left wing $WingL$, extension of the right wing $WingR$, arbitrary wing extension $Wing$ and extension of both wings $WingBi$. Further, wing extension towards the other fly $WingT$ or away from the other fly $WingA$ are classified.

Filtered values are available for all wing classifiers with $Wingcont = 0.5$ [sec].

Figure 59 depicts the $Wing$ classifiers and all their underlying conditions and attributes for the same time series as in figure 58 above. The classifiers WingT and WingA are used to generate figure 73 on p. 172.

Note that the example time series in figure 59 contains wing extension events that are detected during occlusion (see section 5.2.4.2).

Definition 117 *The attribute isOcclusion is defined as true whenever only one fly body region is detected or when* $\frac{Area_A}{Area_B} < 0.1$.

Definition 118 *Condition CopA is defined as* $CopA = isOcclusion$

Definition 119 *The classifier Cop is defined as* $Circ = CopA$ *with* $Copcont = 25$ *[sec].*

Occlusions that last longer than 25 sec are scored as a copulation event.

In order to derive a courtship index, like in section 5.2.5.1, courtship as a whole may be defined as any of its subbehaviours.

Definition 120 *The classifier Crt may be defined as* $Crt = Folled \vee Oried \vee Circed \vee Winged$.

Figure 60 depicts screenshots of the annotationTool that summarize all major events for the time series in the figures 56, 57, 58 and 59.

The resulting classifications may be color coded and visualized as an ethogram.

Figure 61 color codes the time series from figure 60 (a), the assigned colors blue for *Oried*, green for *Folled*, red for *Winged*, resp. particular light red for *WingAed*, dark red (as in figure 61 for *WingTed* and another yellowish light red for *WingBied*), yellow for *Coped* and black for the *isOcclusion* attribute in the last line. In (b) the detected attributes are displayed in a single line, events overcolor each other according to the following ranking: *Coped*, *Circed*, *WingBied*, *WingAed*, *WingTed*, *Folled* and *Oried*, e.g. when *Coped* is true the compact line will be colored in yellow, no matter whether other events would be detected as well.

The detailed view in 61 (a) contains the individual classifiers per line and provides an unambiguous visualization of simultaneously occurring courtship steps, Figure 62 summarizes multiple video chambers in a picture where every single line visualizes the behavior of a particular fly in a video chamber.

The color-ethogram visualization may be used to visually compare the behaviour of different flies. Figure 63 (a) visualizes a wild-type male behaviour, wild-type males are expected to court heavily, while (b) visualizes the behaviour of a wild-type female, it is known that wild-type females do not court. As expected the behaviour visualizations show a clear difference between (a) and (b).

Further, figure 63 (c) and (d) visualizes the behaviours of known mutants. In (c) the behaviour of a fru^F male, a "feminized" male is depicted. As expected, the ethogram is mostly white, much like in (b) and opposed to (a).

Finally, in figure 63 (d) the behaviour of a fru^M female, a "masculiized" female is depicted. The resulting visualization is more colorful than the ones in (b) and (c), but not as rich as the one in (a). This is mainly since the fru^M female is anatomically unable to copulate and receives less encouragement from the other female fly.

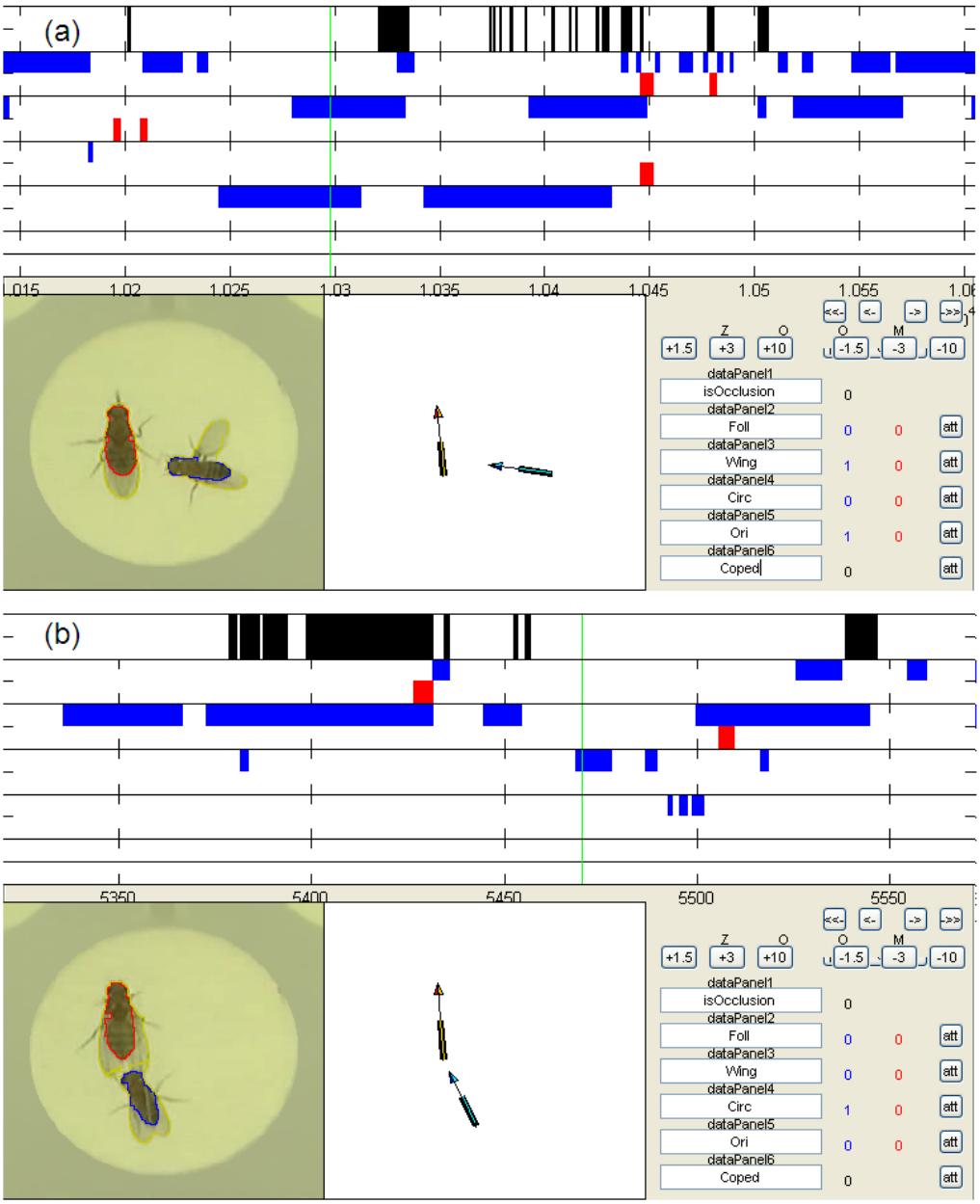


Figure 60: Screenshots of the annotationTool, the green line depicts the position of the displayed picture within the time series. (a) shows picture and summarizes the major events for the time series used in figures 58 and 59. (b) event summary and picture for time series used in figures 56 and 57.

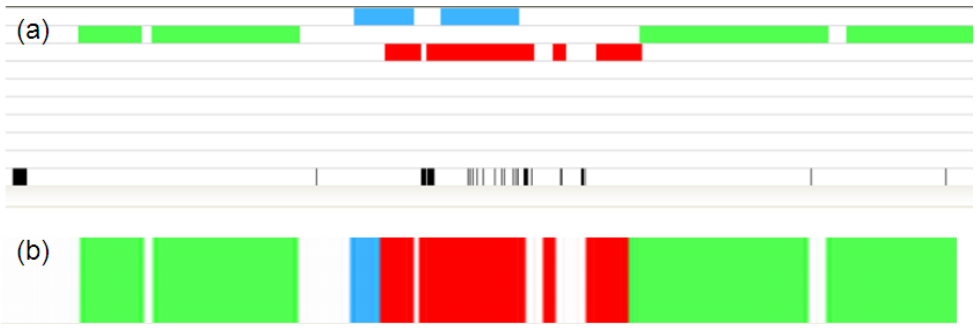


Figure 61: Color coded ethogram (a) detected events per line, in order: Orientation (blue) fly A, Following (green) fly A, Wing Extension (red) fly A, Circling (magenta) fly A, Orientation fly B, Following fly B, Wing Extension fly B, Circling fly B, Copulation. The last line visualizes the *isOcclusion* attribute. (b) compact visualization of all detected events for fly A.

The differences within the ethograms in figure 63 indicate that the top-down defined classifiers are sex specific and fru dependent, see the differences between left and right resp. between top and bottom ethograms.

The defined classifiers enable behaviour comparison in different terms, e.g. the of total time of specific events, latency to the first occurrence of an event, number of observed events, occurrence of events within bins of one minute, percentage of occurrence during the total video length or percentage of occurrence before copulation.

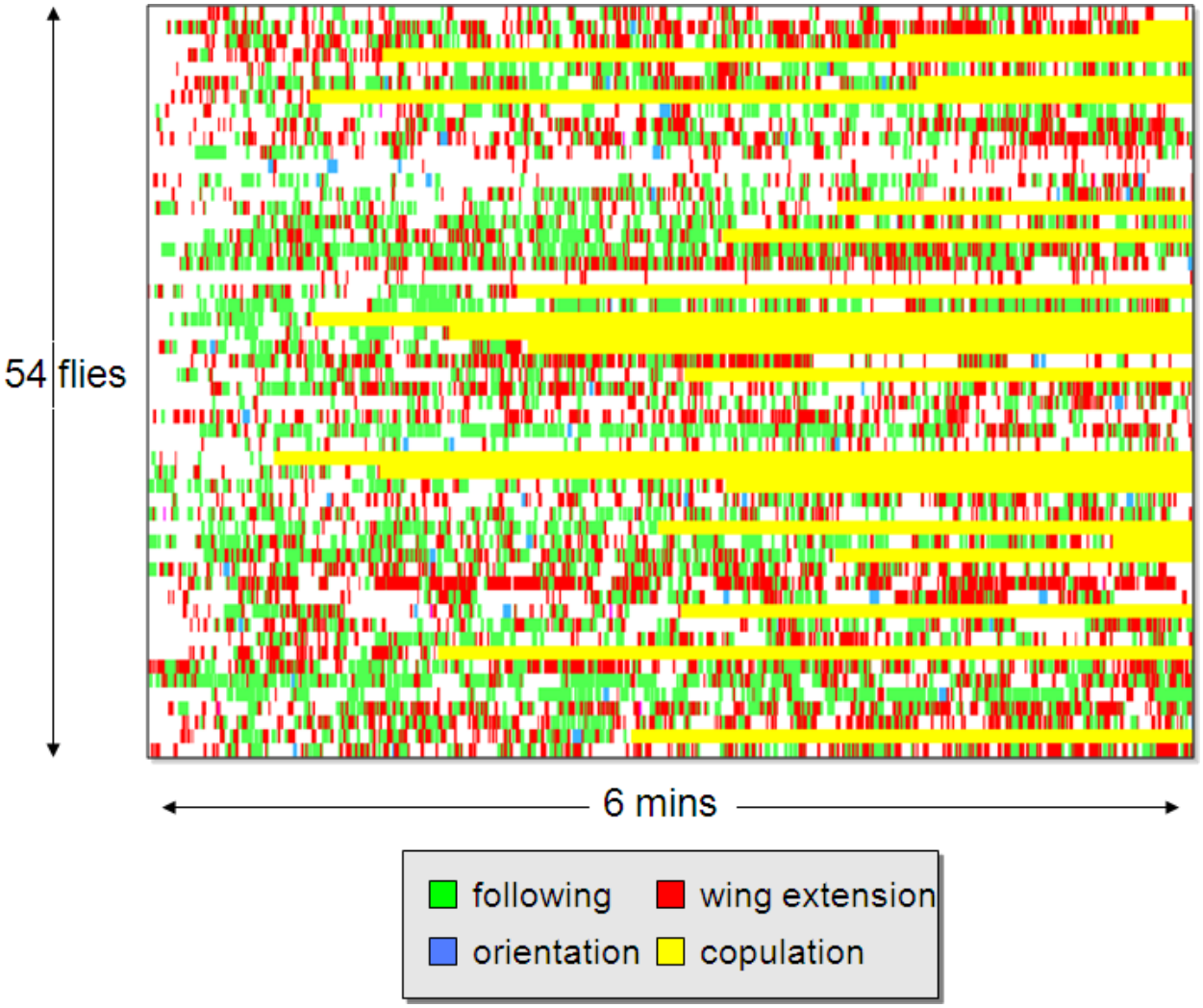


Figure 62: Visualization of multiple chambers in a color-coded ethogram. Each line contains a compact view as in figure 61 (b), time of event occurrence is visualized in x-direction. The legend visualizes the color code.

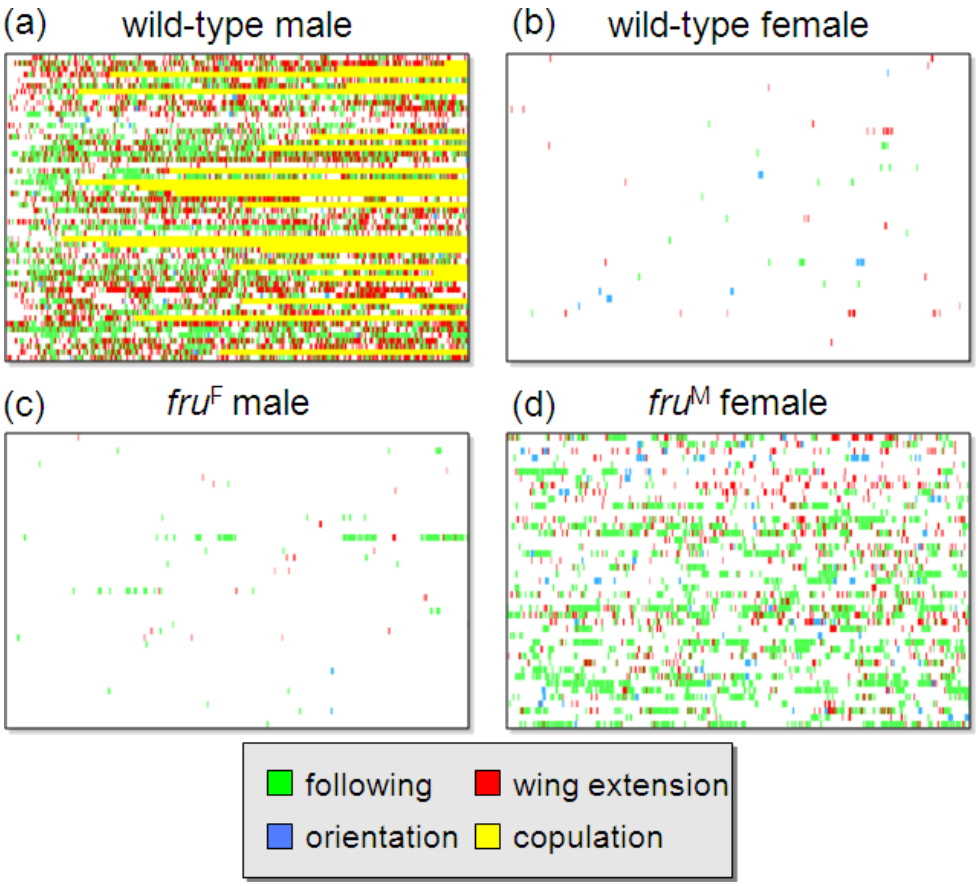


Figure 63: Behaviour visualization of wild-type behaviour and known mutants. (a) wild-type male behaviour (b) wild-type female behaviour (c) Fru^F male behaviour (d) fru^M female behaviour.

6 Tools

6.1 webInterface

The webInterface provides a web-based graphical user interface for processing videos, it hides the complexity of the processing pipeline from an operating user. The webInterface interconnects a number of hardware and software components.

- the computing cluster: a steadily growing set of more than 500 CPU cores that may be used for in parallel for video processing
- the network storage: a giant storage containing videos and processing results, consisting of more than 100 TB of permanent storage of "low availability", which means that it may take up to an hour to access data, and 10 TB of temporary processing space of high availability, which means that data is immediately and fast accessible
- the web server: a fast and highly available server that is connected to all other hardware components and hosts the web interface. Besides managing videos and processing data it also allows to play videos and to invoke the annotationTool (see section 6.2) on a local machine such that to processing data located at the network storage may be inspected and annotated.
- the system components introduced in section 2, in particular in figure 2 on p. 13, consisting of the preprocessor, the tracker, the postprocessor and the annotationTool.
- a number of scripts invoked by the web server, which allow to submit preprocessor, tracker and postprocessor jobs to the computer cluster, but also to upload video data, to convert video formats, to generate processing result websites or to check the processing status of active cluster nodes.

The screenshots below provide an overview of the webInterface. Figure 64 depicts the video upload interface, which allows to import videos for future processing and management into the webInterface resp. its database. Videos may be imported to different *projects*, by default videos are imported to a project named after the user himself. The summary bar in the top provides a summary for all videos within the selected project, videos may be in different status categories, namely "Recorded", i.e. newly imported, "Preprocess", "Tracking" or "Postprocess". Each status category contains stati "Start", which means that the processing task is currently running, "Failed" and "End", which means that the particular processing step has successfully finished. Status category "Postprocess" further contains a status "Incorporated", since postprocessed data may be inspected and annotated by the annotationTool and annotations may then be incorporated into the processing results.

The dialog selection bar on top allows to switch between the video tracking view (see figure 65 or figure 4 on p. 19), the video upload view (see figure 64), a view showing the cluster status[65] and the option to log off.

Before the uploading step the user may copy videos to a specific part of the network storage, each user has write access to a directory named after his username. For the uploading step itself the user may specify a number of videos from that folder and optionally meta-information for those videos which are then imported in to the system. This importing step involves creation of database entries, but also movement of the selected videos to a write-protected part of the network storage. This ensures that videos are not altered while they are processed. Correspondingly, a user may "eject" videos he is no longer interested in and the system will remove the videos from a project. It is not possible to delete data via the webInterface, videos are instead moved back to the writeable network space, from where they may be re-imported or otherwise curated.

Figure 65 (and figure 4 on p. 19) show a list of processed videos, the status column contains yellow entries for videos that are currently being processed, red entries for videos that failed and green entries for videos that successfully finished a processing step and are ready for further processing. For such videos the next a "start" button appears which starts the next processing step.

When clicking the "Reports" link on the right the results of the processing steps so far is displayed on a separate webpage, figure 66 shows a screenshot of this result page after the preprocessing step. The screenshot mainly contains pictures of the different background models and a part of the text output that was generated during runtime. The system further automatically generates hough detection pictures as in figure 10 on p. 34 or ethogram pictures as in figures 61 on p. 155 and 62 on p. 156 per video, all automatically generated pictures and text may easily be linked into the result webpage.

Besides the result page key information is displayed in a extended per-chamber view which opens when clicking the icon below a videos name.

Figure 67 and figure 5 on p. 20 depict a screenshot containing more detailed chamberwise information, in particular the preprocessor generates the two summary pictures on the left, one depicting the watched boundaries of the chambers and one providing an unambiguous legend for the numbering of all detected arenas. Further an acception or rejection picture per chamber is displayed for each chamber (figure 5 contains a rejected chamber) and the processing status per chamber is displayed, in figure 67 the status chamber 11 is "post_failed", and this status is inherited to the whole video. The status of a video is computed from all its child chambers, in case one chamber fails the video is considered to have failed, otherwise the video status is either success or *in progress*, i.e.in one of yellow states, in case one of its child chambers is currently being processed.

[65] The cluster status view displays the output of the *qstat* command and offers a "refresh" button.

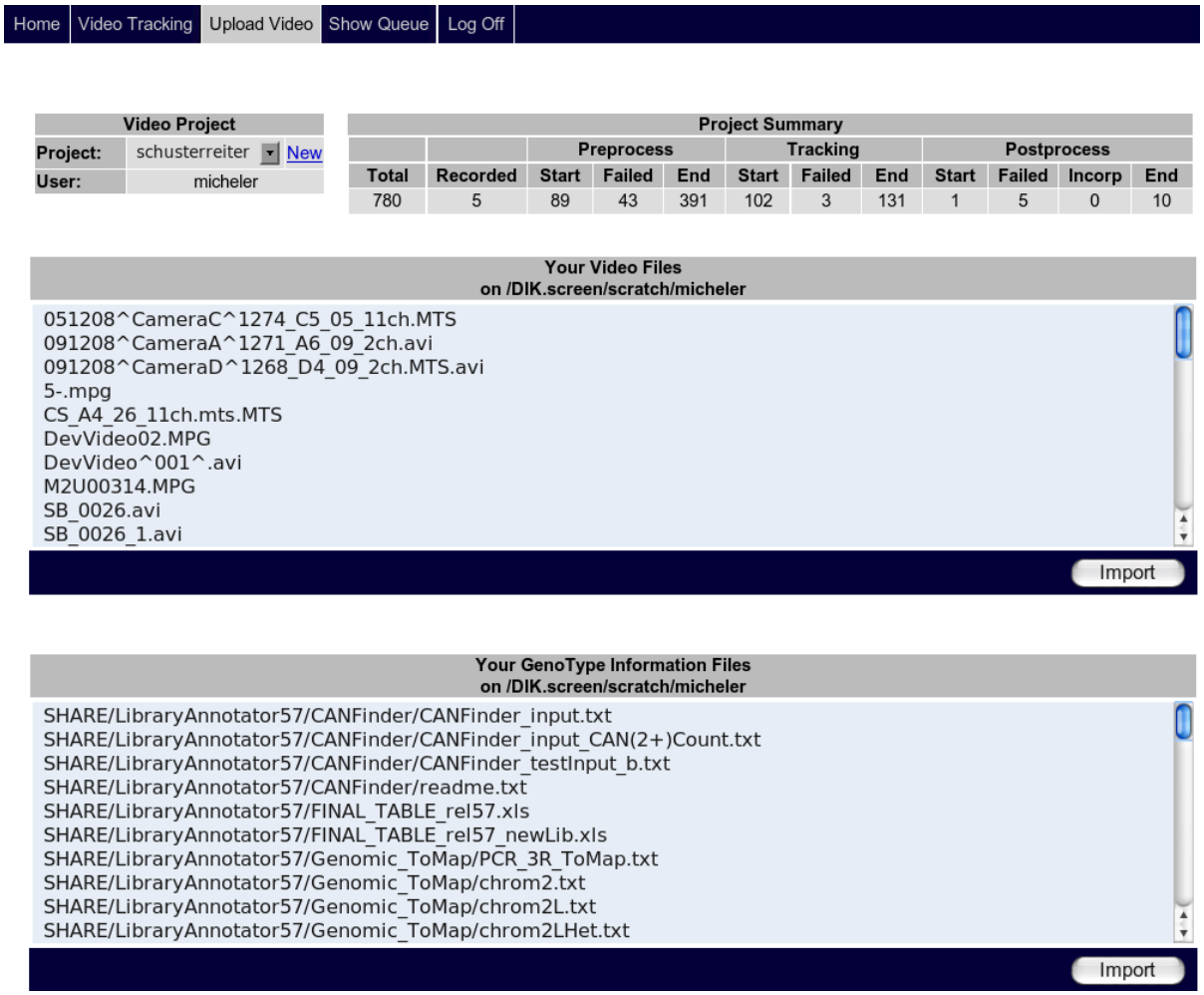


Figure 64: Screenshot of the webInterface showing the video upload mask and a project summary outlining how many videos are at which status.

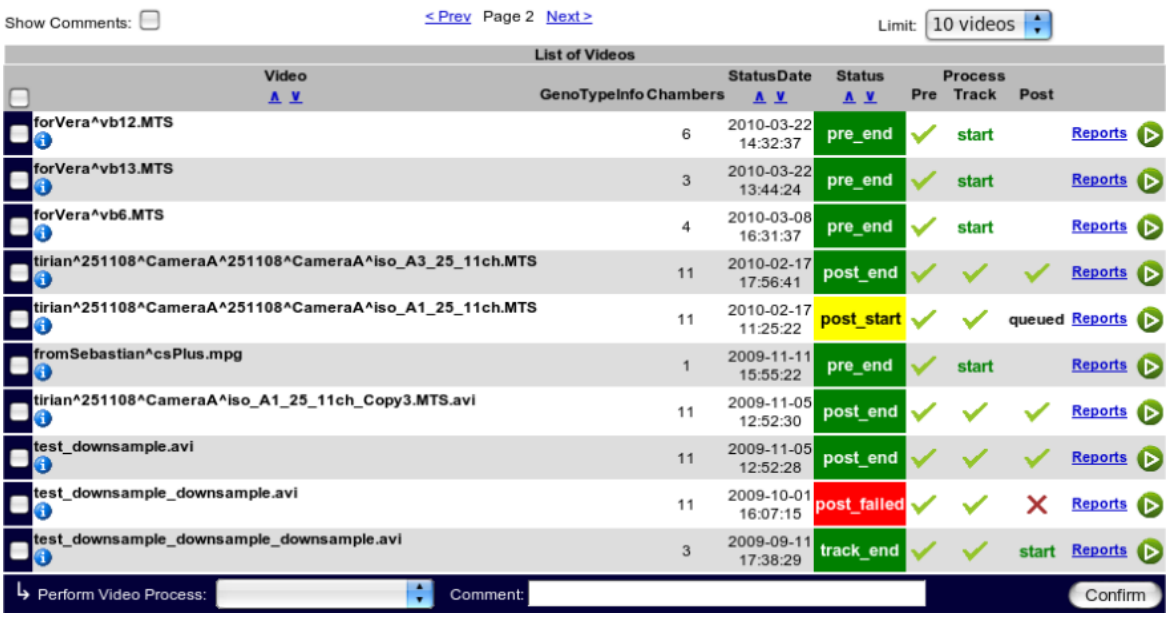


Figure 65: Screenshot of the video list view of the webInterface. The top bar allows to select browse the videos of a selected project, the checkboxes on the left allow to select videos and the dialog on the bottom allows to apply specific operations (like "track", "eject" or "add comment") to the selected videos. The list itself contains the video names, the number of detected chambers, the current status of the video, the date of the last status change and GUI elements to start further processing steps, view processing results or to play the video.

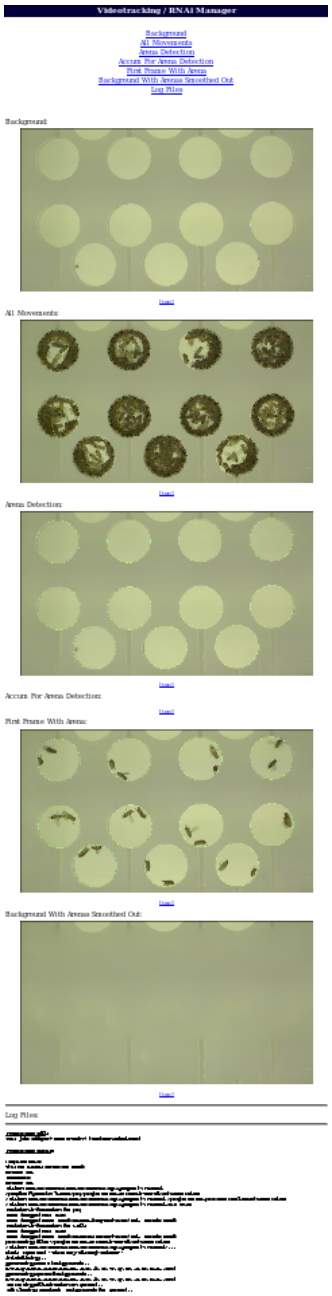


Figure 66: Screenshot of the per-video report after the preprocessing step, depicting the median background, the pixelwise minimum of the frames that were used to compute that background, the cautiously smoothed background model, the first frame of the video overlayed with the detected arenas, the rigorously smoothed background model and the output text produced by during preprocessor runtime.

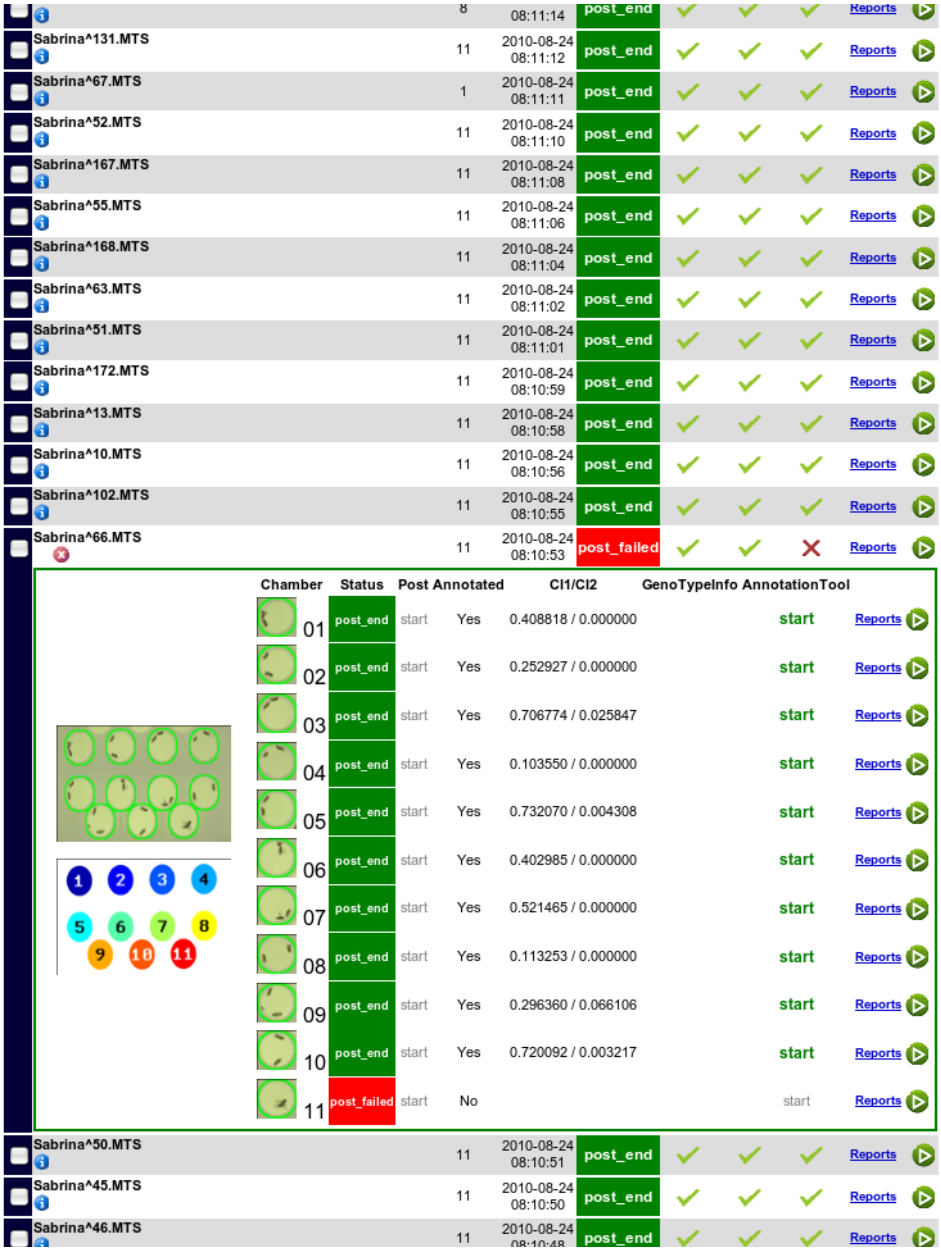


Figure 67: Screenshot of the chamber view of the webInterface, depicting the video summary pictures on the left, per-video acception of rejection pictures, the courtship index summary and GUI elements for further processing, inspection or annotation steps.

The chamber view further contains a summary showing the courtship index for each fly in case it has already been computed, the genotype of the flies in that chamber, in case that information has been provided by the user, and GUI elements for starting further processing steps, for invoking the annotation tool for inspecting per-chamber reports or to play chamber videos[66].

When invoking the annotationTool (see section 6.2) from the webInterface, saving an annotation within the annotationTool will also notify the web server about the newly available annotation information such that the newly saved annotations will immediately be pulled to the read-only network space that also contains all other processing results. Re-postprocessing such an annotated video will cause the system to incorporate given manual annotations into all downstream processing steps, further a special "slim" postprocessing step named "Incorporate", which recomputes annotated results only, may be invoked to update computed courtship indices and ethograms.

The webinterface is provided for multiple versions of the system, e.g. for a development and for a production version. Further some special data sets or the spin offs introduced in sections 6.3.1 and 6.3.2 may require invocation of different modules or of specific versions of the system, the webInterface encapsulates such data sets together with the to-be-invoked system modules, a separation into different instances webInterface prevents confusion of incomparable data sets and processing with an unexpected modules.

6.2 annotationTool

The annotationTool may be used to inspect and annotate data. Figure 68 shows a screenshot of the annotationTool, it consists of six data panels on top, two video panels in the middle and GUI elements on the right side. The title of the annotation window, which is not displayed in the sample figures below, contains video name and chamber number of the currently inspected video.

The data panels may be used to inspect data, the top most data panel always denotes occlusions, black bars mark occluded frames. The five other data panels display either binary attributes, e.g. classifier predictions as data panel five and six, or continuous values as in data panels two to four. The content of the data panels may be selected by the GUI elements on the right, the text boxes may be used to enter known attribute names, the values of the selected attributes in the current frame are written next to the attributes, or the "att" button to the very right may be used to choose an attribute out of a list.

The user may navigate through the data panel by pressing the left and right buttons right below the data panel and he may zoom in or out by using the

[66] Remember, after the preprocessing step the original video is split into single chamber videos.

zoom buttons or by selecting the to-be-displayed range with the mouse. Right-clicking the mouse zooms out such that the whole time series is displayed in the annotation window. The data panel navigation buttons scroll all panels either by a full '->>' or by a half '->' screen without moving the selected *current frame*.

The green line marks the current frame, the frame may be selected either by clicking on the data panels, by entering a frame number into the filed called "frame" or by pushing a "previous" '<', a "next" '>' button or the "play" 'P' and "pause" '||' button, e.g. next to the frame field.

The two video panels contain the original video, optionally with the perimeters of the tracked body regions and wing regions drawn into the video frame, and a comic visualizing the detected regions as arrows. Each arrow is colored either by blue or by red, blue always denotes the smaller fly, which is typically the male, while red denotes the larger fly, typically the female. The arrow points from the *Tail* of the fly to its *Head* along the flies *Orientation*, the transition between filled and slim arrow shaft denotes the flies *Centroid*.

In case the annotationTool depicts an occluded frame (see figure 69), the right video panel depicts the original video overlayed with the interpolated ellipses for the two flies and another video panel showing the occlusion assignment is shown. This panel depicts the last frame before an occlusion and the first frame after an occlusion and connects the re-identified flies with a colored line, in figure 69 the blue line connects the two male flies while the red line connects the two female flies. The user may overrule a machine assignment by clicking the "switch" 'X' button on the very right, this button switches the identities of the two flies from the current frame up to the end of the video.

Besides the per-frame navigation the user may also navigate *per occlusion*, or per *heading-sequence* by using the previous, next, play or "rewind" '|<' buttons. Whenever an occluded frame is displayed the occlusion number is displayed in the bottom-most text field, a user may proceed to the next occlusion, play that occlusion or add an occlusion number to jump to a particular occlusion. In case a non-occluded frame is displayed the field contains the last occlusion number followed by a '+', in figure 68 it is "2+", denoting that the sequence after occlusion 2 is inspected.

The sequence of occlusions may be sorted either by occurrence or by assignment confidence using the radio buttons at the very bottom, the rewind button below the "by confidence" selection selects the least confident occlusion, a user may then use the next button to inspect more confident occlusion assignments, the current confident score is displayed right to the radio button.

The GUI elements for annotating the *heading*s of flies are deprecated, the user may still annotate resp. manually correct the heading for flies within

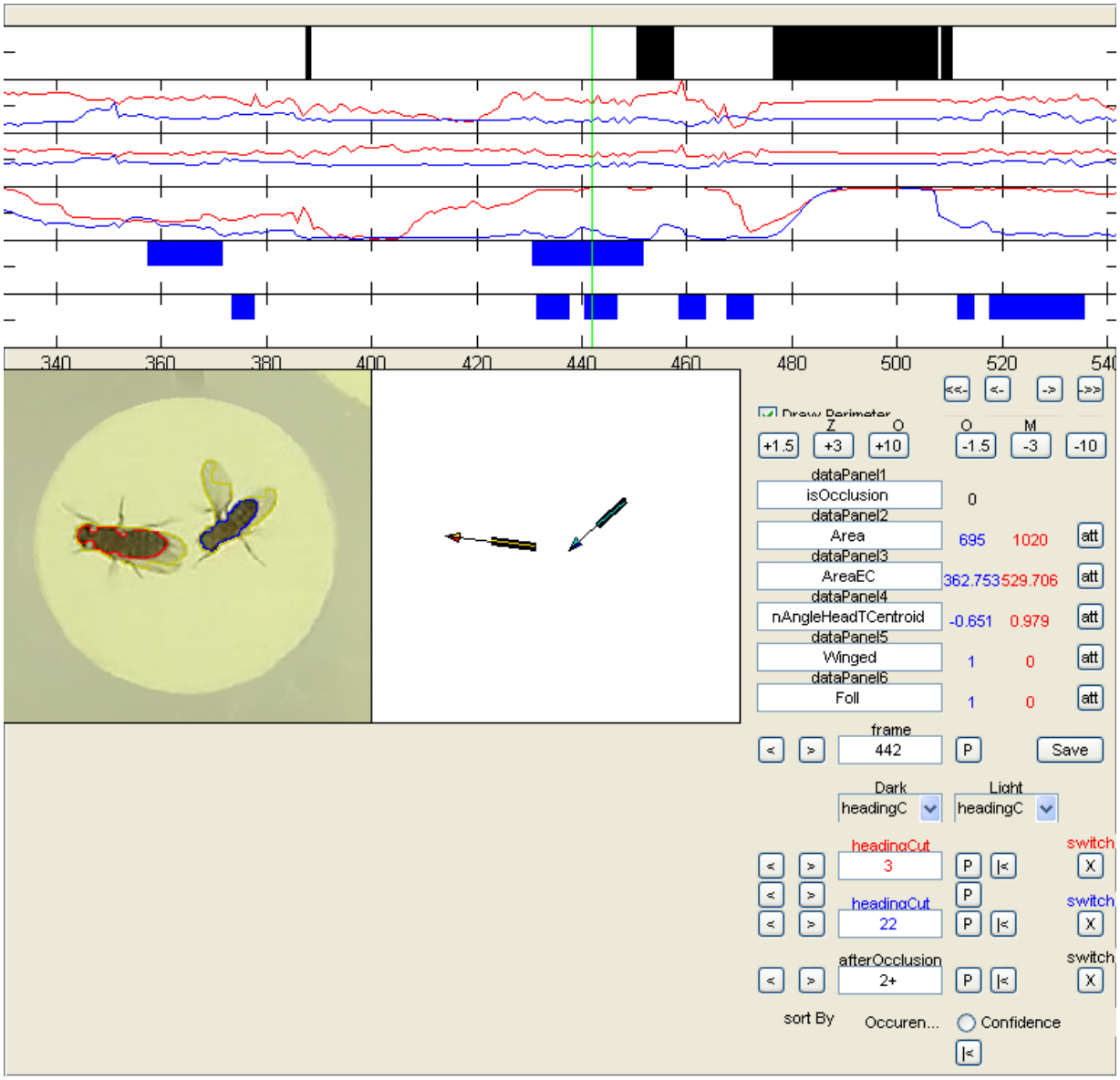


Figure 68: Screenshot of the annotationTool in a non-occluded frame.

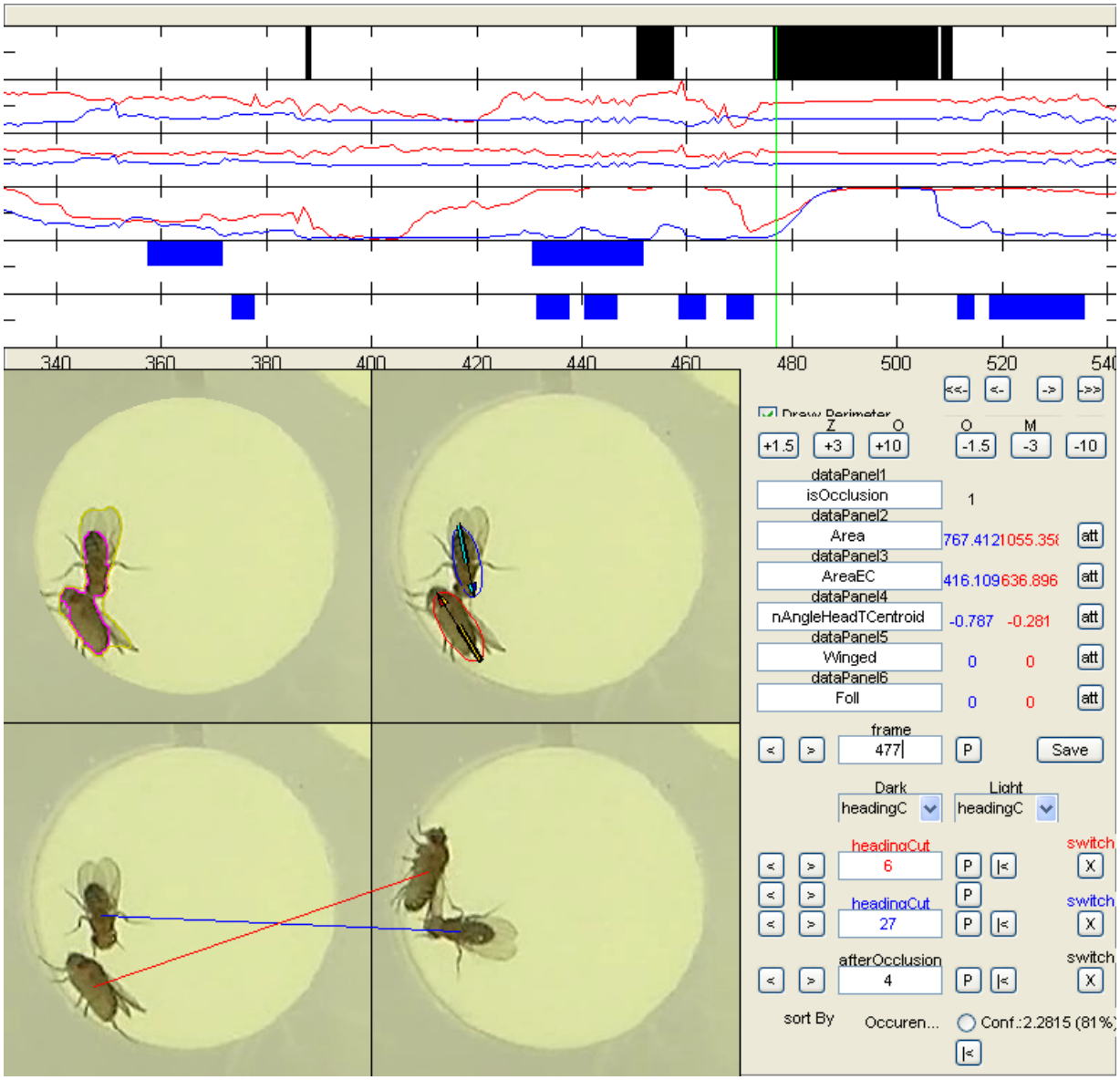


Figure 69: Screenshot of the annotationTool in an occluded frame.

different frames, however, experience has shown that this annotation is not required in praxis.

The user may save the current occlusion annotation by clicking the "Save" button next to the per-frame navigation buttons, in case the annotationTool was invoked by the webInterface saving an annotation will also notify the web server about the newly available annotated data. Note that the annotationTool never overwrites data but generates files tagged with timestamps, such that the newest version of an annotation may easily be determined, but such that also past annotations at given time points may easily be reconstructed. For this reason, and since the system's binary files follow the same principle, past processing results may be reconstructed from given videos.

6.3 Spin Offs

6.3.1 Spin Off 1: Chaining Detector

The modular architecture of the system allows it to be utilized for further research questions in addition to the analysis of courtship behaviour. One spin off, built upon an early branch of the system, was designed with the aim to detect fly *chaining*, a behavior where flies stay close together and follow each other.This derived variant of the system required an adaptation of the preprocessor that detects arenas by brightness. For this purpose a standard median background model was sufficient, there was no robustness to dirt and therefore no smoothed background required and only fly body regions were to be detected during the segmentation step. Furthermore this experimental assay does not require to solve occlusions as the behavior is quantified as an average of all flies within a chamber and the group of flies within the chamber is scored as a whole rather than each of its individuals.

The experimental setup depicted in figure 70 (a) was designed to analyze male chaining behaviour in presence of a fly courtship song, it has been observed that male flies perform this chaining behavior in presence of the song. Figure 70 (b) shows in yellow the manual annotations of a biologist (yellow and cyan) versus a computer generated approximation measure (red and blue). During the middle of the plot the courtship song was "turned on" and a recorded courtship song was played through a speaker box. The plot shows that both, manual and automated quantification see a significant change in behaviors with respectively without the courtship song being played. While the manual annotation perfectly captures both chaining and non-chaining behavior, the automated method shows a less, but still very clear difference. The difference between the quantification methods is explained by the use of a simplified measure and by human bias. The machine does neither filter random encounters in the first half nor ignore individual non-chaining flies in the second half.

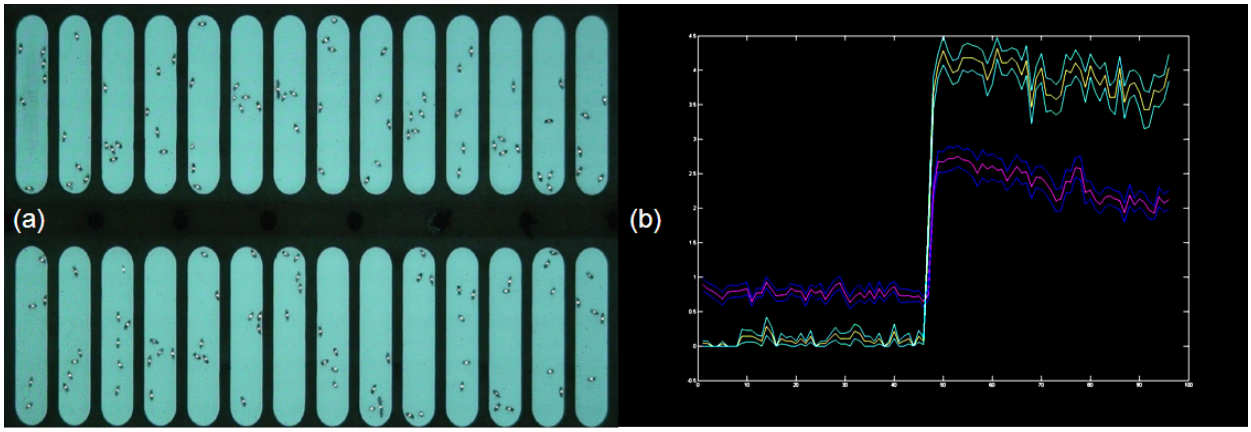


Figure 70: (a) snapshot of the experimental setup where the spin off variant aimed to analyze. Flies perform in bright chambers and eventually show chaining behavior (b) manually annotations marked in yellow versus an automatic approximation measure marked in red and blue. The sharp rise in the middle happens at the time point when the courtship song was turned on. Both, manual and automatic assessments capture a significant change.

6.3.2 Spin Off 2: Food Preference Analysis

A later spin off is applied to analyze foraging behaviour experiments. Recorded arenas contain food spots in blue (yeast) and red (sucrose) and the setup aims to analyze foraging and food preference behavior, more specific to measure the time flies spend at or close to yeast or sucrose nutrition. The spin-off required a refinement of the preprocessor as recorded arenas contain food spots. Figure 71 shows a snapshot picture of the setup and depicts how individual food spots are detected and discriminated by color. In particular (a) shows the experiment setup where the food spots are visible, food spots in blue contain yeast and food spots in red contain sucrose. Subfigure (b) shows the results of an additional food spot detection step, detected food spots regions are drawn in different colors. The food spot color in (a) is used to automatically discriminate the different types of food.

The tracking module was reused, but a simple median background model was sufficient as robustness to dirt or dealing with static objects was required, and again only fly body regions were of interest for downstream analysis. Multiple, about twenty flies were tracked at the same time and behavior was quantified as an average per chamber, such that again occlusions were not required to be solved for downstream analysis.

The spin-off's postprocessor analysis allows to quantify the amount of time that flies spend at or close to specific food spots. Figure 72 (a) shows the voronoi regions for each food spot, which contains all points that are closer to that food

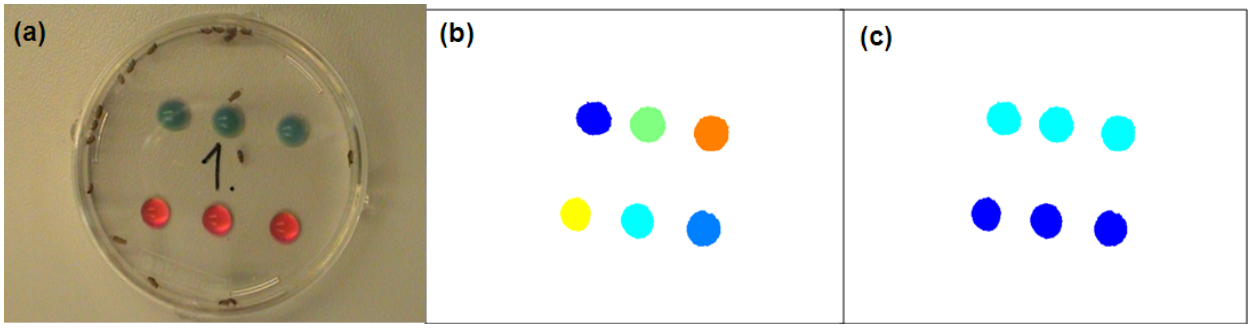


Figure 71: Preprocessing steps handling food spots (a) snapshot of a sample video. (b) detected food spots visualized in different colors. (c) food spots grouped by detected food color.

spot than to any other food spot, the partitioning of the arena in different food spot zones enables to analyze where flies prefer to stay while they are not feeding. Subfigures (b), (c) and (d) show how more advanced foraging analysis may look like, (b) quantifies feeding of different food sources, (c) quantifies feeding at individual food spot while (d) quantifies the closest food spot over time, regardless whether flies are feeding or not.

The flies in the video summarized in figure 72 prefer sucrose over yeast and spend more time at sucrose food spots. However, they also spend lots of time close to yeast food spots but do not feed from those. Flies were mostly detected in voronoi regions of the outer food spots, this may result from flies exploring the arena border. Computing a separate voronoi region for the border allows to specifically capture flies that are closer to the border than to a food spot.

Such a border voronoi region may be derived by treating every border pixel as a voronoi center and then unifying all resulting border pixel voronoi regions into a single border voronoi region.

Figure 72 shows an automatic quantification for the assay introduced in [22], the necessary adaptions to the postprocessor modules were implemented very quickly (see Acknowledgments).

6.3.3 Spin Off 3: Analysis of Wing Extension Direction

The last spin-off presented is a natural extension of the standard courtship quantifying system. The analysis of this spin-off goes deeper into details and further quantifies the position of the female fly during wing events

Figure 73 visualizes the *relative female positions* during wing extension events. For each frame where a wing extension event was detected the position of the female is determined. All these scenes are then translated and rotated

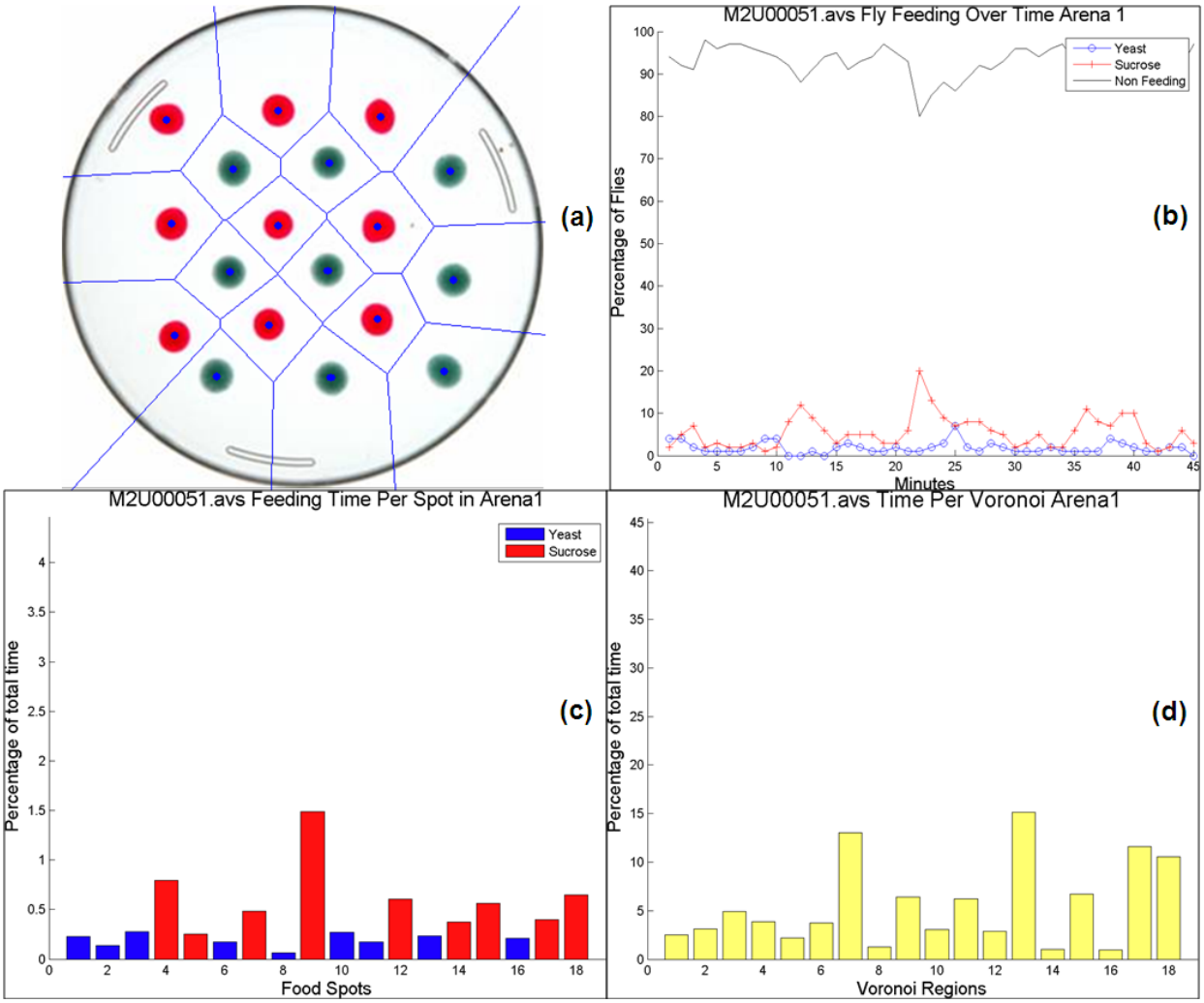


Figure 72: Results of a sample analysis. (a) shows the voronoi regions for each food spot, a food spot surrounding partitioning of the arena. (b) quantifies the percentage of flies per food type over time (c) quantifies the percentage of time that a fly spent directly on a particular food spot, normalized by the total number of flies (d) quantifies the percentage of time that a fly spent in the voronoi region of a particular foodspot, normalized by the total number of flies but *not* normalized by the size of the voronoi area. Subfigures (b), (c) and (d) were computed by Lorenz Pammer.

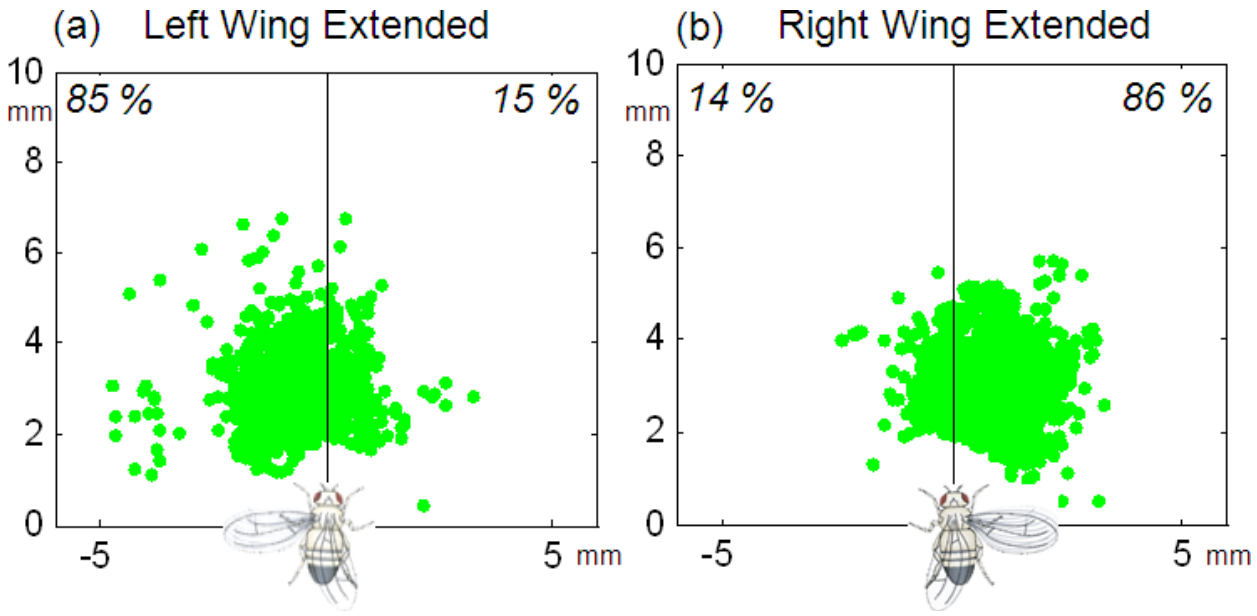


Figure 73: Wing Extension Direction Analysis. (a) depicts the female positions when the male fly extended its left wing, (b) depicts the female positions when the male fly extended its right wing.

into a coordinate system where the male fly is at the origin heading north, and the female position relative to the male position is marked by a green dot.

Interestingly, the female fly seems to be to the left of the male fly whenever the male fly extended its left resp. it is to the right when the male fly extended its right wing. The observations in figure 73 strongly suggest that the male fly extends its wing towards the female fly, the female fly is in about 85 % of the analyzed wing extension cases at the same side where the wing is extended to. Note that the male fly sometimes extends both wings in case the female is straight ahead, and that, in case of a change in the relative female position, an adaption of wing extension direction is sometimes difficult resp. slightly delayed due to physical aviation limitations.

However, the quick adaption in wing extension direction suggests that visual queues are involved when a male fly decides which wing to extend, rather than or in addition to gustatory queues that are believed to be involved here. Given videos from specifically designed experiment series the wing extension direction analysis toolkit may be used to analyze sensory information integration in flies.

7 Conclusion

This thesis introduced a software bundle for automated analysis of Drosophila courtship behaviour which is designed to minimize user interaction and enables robust, objective and high-throughput analysis of courtship videos.

During its analysis the system technically *densifies*[67] biological data in two steps: an image processing step that translates a video into a time series (see chapters 3 and 4) and a pattern recognition step that searches for known or learned patterns within that time series (see 5.2.5). Many methods described within this thesis contain data transformation or data cleaning operations at image level (see e.g. sections 3.2.2, 4.2.2 or 4.2.3) or at time series level (see e.g. sections 5.2.3.1, 5.2.1 or 5.2.3.3) that finally enable a simplified but rich characterization of the observed behaviours as result ethograms (see section 5.2.5).

Figure 74 visually summarizes the biological context for the system's application. Subfigure (a) depicts the brain of a fruit fly Drosophila melanogaster, a well-established model organism in the field of neuroscience. By application of rigorous available genetic tools biologists are able to modify single or small groups of neurons within a fly brain, the aim of such manipulations may be to understand how innate behaviours are encoded in genes resp. how genetic circuits produce hard-wired neuronal circuits, and how these neuronal circuits correspond to certain behaviours. Besides the fly brain subfigure (a) depicts a small set of neurons that are targetable resp. may be manipulated by available genetic tools.

Subfigure (b) depicts a screenshot of a courtship video, where either wildtype flies or flies with manipulated neurons may be recorded in order to study a circuits *functionality*.

Besides functional studies a neuronal circuit may also be studied by its *anatomy*, subfigure (c) depicts a schema of a neuronal circuits and highlights that flies with manipulated neurons may have differently hard-wired neuronal circuits.

The system introduced in this thesis may be used to quantify the behaviours of flies in videos (b), subfigure (d) depicts a change in ethograms resulting from a manipulated circuit. Perturbations of neurons and the quantification of resulting changes in behaviour may therefore be used to systematically reconstruct the functionality of neuronal circuits and the neurons they are composed of resp. to identify involved genes or neurons.

When applying the system the primary goal is to filter out interesting phenotypes from a large set of videos, such that it does not miss interesting positives.

[67] The system translates high amounts of video data that has low information content and into low amounts of time series data containing biologically relevant attributes of high information content.

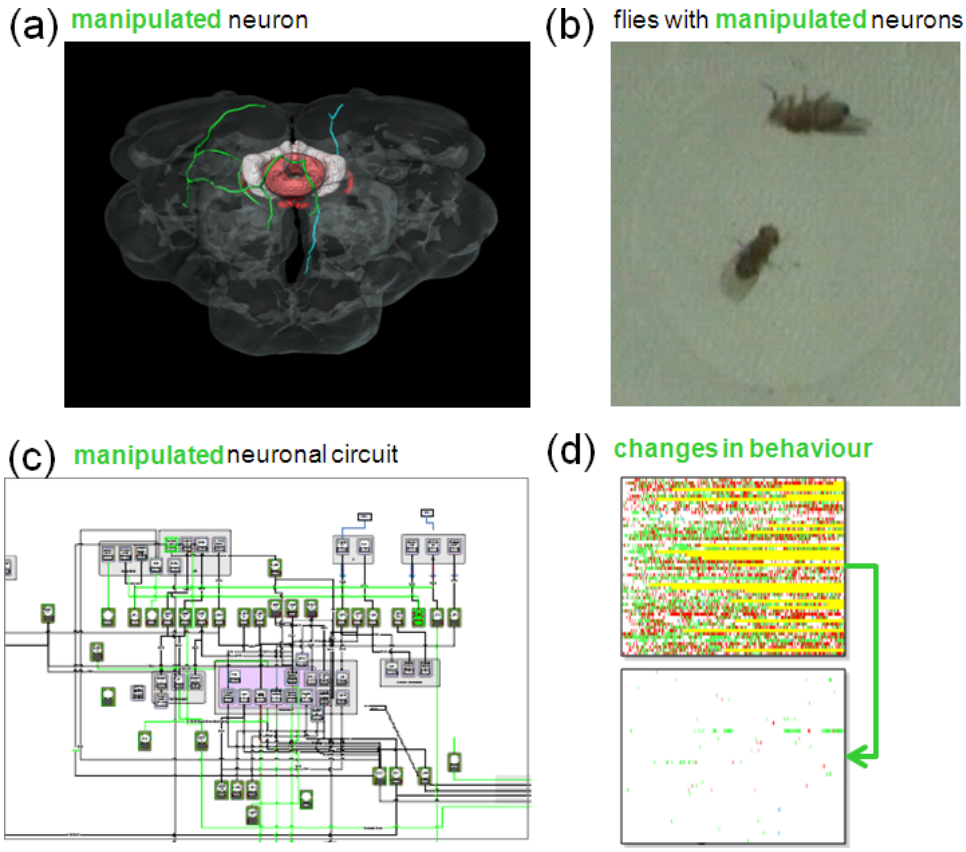


Figure 74: Schematic visualization of the biological context for the system's application. (a) biologists are able to modify single or small groups of neurons within the fly brain, the picture shows a schematic fly brain and some genetically targetable neurons (b) flies with manipulated neurons are recorded in videos. (c) flies with manipulated neurons may have differently hard-wired neuronal circuits, the picture shows a schematic neuronal circuit. (d) the behaviour in videos containing such flies with manipulated neurons resp. manipulated circuits are quantified by the system introduced in this thesis, the picture shows a change in ethograms resulting from a manipulated circuit.

False alerts may be detected by reproducing videos with same genotype, one reason for choosing Drosophila as a model organism is that one may easily produce large numbers experiments, especially for flies of same genotype.

In order to enable dealing with large numbers of videos no system configuration or manual tweaking of parameters required when handling the system, manual user interaction is supported but typically not required. The only requirements are that the user specifies the arena diameter[68] for normalization purposes and that the videos are conform to the quality control standards.

In order to pass the preprocessor quality control a video has to be conform to the following quality standards:

- exactly 2 flies per chamber are expected.
- Arenas may neither be moved nor intruded.
  Arena detection relies on non-moved movies, and arena infringement may affect the fly behaviours.
  Besides that, arenas may be come in arbitrary number and positions and there is no particular contrast required between arena and non-arena background.
- All arenas are assumed to have same size and must neither overlap with the video border nor with each other.
- Arenas must be surrounded by a uniform background
- It is recommended to remove the loading bars when recording from the video as it may disturb the arena detection step, it is particularly not allowed to place the loading bar such that it enables flies to hide in non-recordable zones.
- It is strongly recommended to turn off auto focus and auto white balance settings of a camera, in case they are available they are typically on by default, and to check that recording settings do not distort the captured video. In case a camera and a recording software is used both aspect ratio settings have to agree.
- Even background lighting is strongly recommended, reflections or mirror effects are disallowed. Lighting should come directly from above or below to minimize shadow effects.
- Video may be compressed in a format that is readable by ffmpeg library for Windows *and* Linux[69], however, compression always comes with artefacts that may potentially disturb image processing steps.

[68] It is sufficient to specify the arena diameter once for all videos aggregated within a project.
[69] Linux readable format is required for processing on a Linux-based computer cluster, in case data is processed only locally, a Windows readable format is sufficient.

Videos that are not conform to the quality standards will be automatically detected and rejected from scoring. Besides the quality control step (see section 3.2.4) the preprocessor extracts necessary background and arena information for the tracker module (see sections 3.2.1, 7 and 3.2.3).

The tracker module segments the fly body and wing regions (see sections 4.2.1.1 and 4.2.1.3) and merges body regions (see section 4.2.2) resp. splits wing regions (see section 4.2.3) if required. As its last step the tracker module extracts the flies primary attributes (see section 4.2.4), which finally translates the given video information into a time series..The preprocessor and the tracker module form the image processing part of the system.

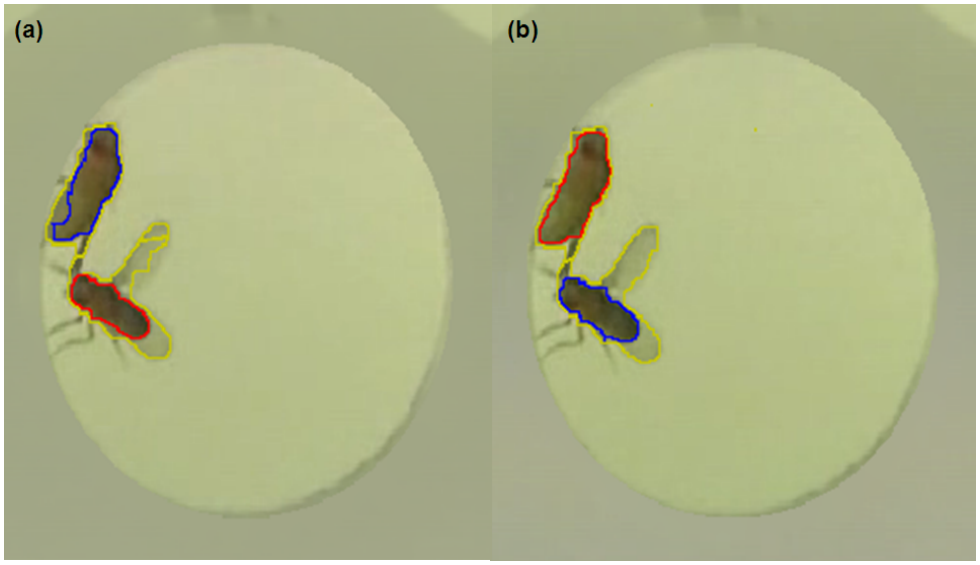


Figure 75: Tracker improvement over time. (a) older version with thresholding and voronoi separated wing areas (b) newest version with gradient corrected thresholding and floodfill separated wing areas.

Figure 75 depicts the improvements of the tracker across several versions of the system, subfigure 75 (a) shows segmentation result of an older version, subfigure 75 (b) segmentation with the current version. Four things are obviously noticeable: The old version did not apply the gradient correction method (see 4.2.1.3), therefore the a part of the stomach of the female fly is not correctly segmented. Further, the old version separated wing regions by the voronoi method while the newer version uses the more suitable floodfill separation (see 4.2.3 p. 69). Furthermore, when taking a close look at the wing area, the vibrating wing is also detected in a better way by the newer version of the system. And finally,

the new version always marks the (smaller) male fly in blue and the female fly in red, while the old version marked the left fly of the first unoccluded sequence in blue (see 5.2.2), assigning male and females to the same identifiers across different videos simplifies aggregated video analysis.

The thresholding based segmentation of the foreground may in principle result in regions of arbitrary size and shape, the extracted primary attributes may be used to reject suspicious segmentations (see section 5.2.3.1). When evaluating different segmentation strategies a point distribution model based approach called *Active Shape Models* [23], which fits a given model with known modes of variation to a given picture and therefore may come with guarantees regarding segmentation sanity, was examined and discarded for performance reasons. The finally chosen method *gradient*2 came with the best results for given time and robustness constraints.

The tracker module is the performance critical part of the system, it translates large amounts of videos data into a time series containing primary attributes and therefore hides the complexity of image processing from downstream analysis modules.

The postprocessor module derives advanced attributes from the simple information basis containing primary attributes only and searches for meaningful patterns within these attributes to derive a few biologically relevant numbers, the events (see section 5.2.5). Such events are summarized into result ethograms, which contain a rich characterization of the behaviours observed within the videos.

Assigning identifiers to flies throughout the entire videos (see section 5.2.1 and in particular section 5.2.2) turned out to be a key challenge for the postprocessor module. The occlusion problem was finally modeled as an optimization problem, the SDH-algorithm (see algorithm 67 in section 5.2.2.2) uses as a dynamic programming approach and therefore guarantees the optimal solution within linear time. The algorithm was further applicable to resolve heading which turned out to be reducible to the solved occlusion problem (see section 5.2.3.3).

Although the system is specialized to analyze courtship behaviour, it provides a toolkit that allows to easily specify other behaviours of interest. It allows top-down (see section 5.2.5.2) or bottom-up (see section 5.2.5.1) specification of behaviours, i.e. rules may either be explicitly defined or machine learned from sample assessments. The annotationTool module (see section 6.2) turned out to be particularly useful for new behaviour specifications as it allows to visualize individual scenes, their machine assessments and all involved observables, including underlying data attributes from the extended time series.

The modular structure of the system further enables the development of spin-off versions, section 6.3.1 describes a chaining detector spin-off, section 6.3.2 a spin-off that analyzes foraging and food choice.

The overall system and its modules are easy to operate by a webInterface which is able to launch processing of thousands of jobs with a few mouse clicks and summarizes result courtship index tables or pictures at a glance or detailed in excel sheets.

The system introduced in this thesis provides an automated and thus robust, objective and high-throughput assessment for courtship videos that allows to analyze the functionality of neuronal circuits and to identify involved genes or neurons.

Appendix: Preliminary Definition and Notation

Mathematically, a video v may be seen as a 4-dimensional construct, $v \subset \mathbb{N}^4$ with dimensions l, h, w and t. It consists of l 3-dimensional frames with height h and width w, a frame $I = h \times w$ further contains color information c for every pixel at position $(x, y) \in h \times w$. In case t is 1-dimensional, $dim(t) = 1$, the value of t is interpreted as a gray value a, $c = a$, in case of $dim(t) = 3$, c in interpreted as a vector $c = (r, g, b)$ consisting of color values for red, green and blue color components. All color values a, r, g, b have values in $D = [0..255]$[70].

Definition 121 *Given a color space D, a color value c consists of a value $a \in D$ or $(r, g, b) \in D^3$.*

Definition 122 *A video v is a set of l frames $v = \{I_f | f \in 1..l\}$, a frame I_f is a set of $h \times w$ color values, $I_f = \{I_f^{x,y} | (x, y) \in h \times w, I_f^{x,y} = c_{f,x,y}\}$, all color values within a video have same dimensionality, $dim(c_{f,x,y}) = dim(c_{f',x',y'}), \forall f, f' \in l, (x, y), (x', y') \in h \asymp w$.*

For ease of notation, the terms region and area and basic binary morphological operations are defined below.

Definition 123 *A pixel is a tuple $P = (p, c)$ consisting of coordinate information $p = (x, y)$ and color value c.*

Notation 124 *Pixel references via coordinates, $I_{x,y}$, $I_f^{x,y}$ refers to the color values of the pixels, references to whole tuples or sets of tuples refers to the coordinate information.*

Definition 125 *an area is a set of connected pixels*

Definition 126 *a region is a set of areas $R = \bigcup_i A_i$. $R = \{(p_i, c_i)\}$,*

arbitrary pixels

Notation 127 *binary region may be denoted as a set enumerating all coordinate information p for pixels with color information $c = 1$, $B = \{p_i | (p_i, c_i) : c_i = 1\}$.*

Definition 128 *Let R be a binary region. The morphological erosion $e(R)$ is defined as a pointwise operation for each point in $p \in R$, such that each point in $e(R)$ is the minimum of all points in its 8-connected neighborhood.*

Definition 129 *Let R be a binary region, $d(R)$ the morphological dilation of definition 30 on p. 69 and $e(R)$ the morphological erosion of region R. The morphological opening of R is defined as a dilation after an erosion, $o(R) = d(e(R))$.*

[70] Binary Images have values in $D = [0..1]$.

Definition 130 *The morphological closing of a binary region R is defined as an erosion after a dilation, $c(R) = e(d(R))$.*

Notation 131 *The occlusion assignment for fly A before the occlusion is fly A after the occlusion and consequently fly B before the occlusion is fly B after the occlusion may be denoted as $AB \rightleftarrows AB$, the assignment for fly A before occlusion is fly B after the occlusion and consequently fly B before the occlusion is fly A after the occlusion as $AB \rightleftarrows BA$.*

List of Figures

1 Snapshot of a courtship video. The video contains several courtship chambers (green), each containing a male and a female fly (blue resp. red arrows). Both flies morphologically consist of two regions, a dark body region and a brighter wing region (black resp. gray arrows). 8

2 The black box contains the main system components (depicted by colored polygons), major workflows (depicted by the arrows) and component interfaces (depicted by matching polygon shapes). Each component (preprocessor, tracker, postprocessor, annotationTool) consists of a bundle of component specific methods. The symbols outside the black box resemble the system interfaces, the system takes videos as input and generates a summary of biologically relevant information. 13

3 (a) symbolizes the background extraction step and depicts the naive median background (b) symbolizes the illumination and color correction step and depicts the illumination correction curve (c) symbolizes the movement detection step and depicts a picture from a rejected, moved chamber (d) symbolizes the arena detection step and depicts the accumulated peaks for circular arena detection (e) and (f) symbolize the background smoothing step, (e) depicts the rigorously smoothed background used for body segmentation (f) depicts the cautiously smoothed background used for wing segmentation (g) symbolizes the border intrusion check step and depicts the watched boundary for each detected arena (h) symbolizes the summarized quality control steps by depicting reasons for rejecting chambers, which are (left to right) too few flies per chamber, too many flies per chamber, chamber boundary intruded by object from outside, chamber was moved (i) symbolizes the arena splitting step and depicts the individual chambers and their watched boundaries without further background in between. The black arrows mark direct dependencies within the information flow, the numbering of subfigures (a)-(i) denotes the processing order. 17

4 Screenshot of the webInterface showing a list of videos in different processing stati. A click to the *start* field initiates the next processing steps, a click to the *Reports* link provides details for the processing so far. 19

5 Screenshot of the webInterface showing per-chamber details. Two summary pictures on the left depict the watched boundary and chamber numbers, each chamber row starts with a quality control acception or rejection picture. Chambers with in a yellow state are currently being processed, green states indicate that processing is done. Clicking a green *start* field in column AnnotationTool invokes the annotationTool with video and tracking information for the selected chamber, already annotated chambers are marked with a *Yes* in column Annotated. 20
6 Gallery of Background Models. (a) median background bg_D. (b) rigorously smoothed background bg_{SA}, used for fly body segmentation. (c) cautiously smoothed background bg_{sF}, used for fly wing segmentation. (d) Pixelwise maximum of frames in $D\prime$. (middle) Visualization of arena numbering, left to right then top to down. 25
7 (a-c) Heatmap of gray values from different perspectives. (a) from top left view (b) view from top (c) view from side (d) resulting compensation curve. 28
8 Result of illumination and color correction. (a) original video frame. (b) illumination and color corrected video frame. 30
9 left: two lines in image space. right: hough transform into parameter space, the axis are r, the distance from the origin and its angle θ. source: [12] 32
10 (left): accumulated votes in hough space as a heat map.. Top picture shows hough space from top and visualizes hough peak positions, bottom picture further visualizes hough peak heights and relative heights. (right) original background picture with detected circles. Circle midpoints are marked with a red cross, circles are drawn in blue. 34
11 (a)-(d) Median backgrounds and detected arenas of four different recording setups 35
12 (a) Kandinsky Composition VIII, 1923. The upper left corner (blue rectangle) is used for further processing (b) filtered circle candidates in blue, midpoints in red. Overlapping and concentric circles are filtered and removed. (c) all circle candidates are drawn, some have weak evidence only. 37
13 Snapshots of chambers with zero, one, two and three flies. The yellow cross marks rejected chambers. 39
14 Witness frame for a moved video, the chamber is rejected and therefore marked with a red cross. 41
15 Overlay of all fly movements within a video. Obviously a fly crossed crossed the scene, and interfered with chambers 3, 7 and 12. 42
16 Snapshot of a video with detected arenas and watched boundary rings (in green). 42

17 Samples for detected boundary intrusions, rejection images contain the witness frame marked with a red cross. 43
18 (a) rescaled background with detected circular arenas. (b) original distorted video frame with elliptic arenas. 44
19 (a) Hough space of a circular hough transform, applied to a background containing sharp black border artefacts. (b) Smoothed background disturbed by blurred cyan border artefacts. 44
20 Arena detection samples, green circles depict correct and expected results, blue circles mark samples with undesired affects. (a) Background with non-rectangular arena placement. (b) Arena overlapping with video border. (c) Forgotten loading bar that affects arena detection. (d) Light reflection that affects arena detection. 46
21 Direct Thresholding. (a) gray value image of original frame, gray values are given as values between 0 and 255. (b) gray value histogram of the image in (a), the height of a bar corresponds to the number of pixels with a particular gray value. The Otsu-threshold is marked in red. (c) binarized foreground region, segmented with method *direct*. 50
22 HSV color space as a color cone. Each color consists of a hue component H, a saturation component S and an intensity component V, source: [15]. 51
23 Thresholding Method. (a) original video frame. (b) foreground of video frame in (a), computed by background subtracting the smoothed background model bg_{SA}. (c) left: gray level histograms of the foreground image in (b), right: gray level histogram smoothed by a moving average filter with window length 25. (d) thresholded, binarized foreground region B, which contains the fly bodies. 53
24 Segmentation of wing regions L and W. (a) original video frame as gray value image. (b) background subtracted foreground image. (c) intensity adjusted foreground image with enhanced contrast. (d) binarized wing region. (e) wing region L is the unification of (d) with body region B. (f) wing region W derived by morphological opening and closing of L. 55
25 Segmented frames containing the original video frame, body region B (red), wing region W (blue) and L (green). The smoothed gray value histograms are depicted below, with body threshold (red) and wing threshold (green). (a) sequence of successive video frames, having similar histogram shapes. (b) alternate frames, showing different histogram shapes. (c) same frames as in (a) directly above, but method $fast2$ instead of $gradient2$ is used to determine thresholds. 57
26 Gradients and edges of fly regions. (a), (b) gradient images X and Y in x resp. y direction. (c) gradient magnitudes G depicting the edges. (d) contrast enhanced gradient magnitudes, bright regions mark borders of fly regions 60

27 Segmentation results for video samples recorded with different brightness settings. (a) Camera setting for brightness set to plus 2 (maximum value). (b) Camera setting for brightness set to minus 2 (minimum value). 62
28 Workflow of segmentation method *gradient2*, black arrows indicate direct dependencies. (a) gray values of an original frame of a single chamber video. (b) background bg_{SA} is subtracted for body region segmentation. (c) symbolizes the threshold determination step, it depicts the gray value histogram (black line) the smoothed histogram (blue line) the body region threshold (red line) and the wing region threshold (green line). (d) binarized fly body region B. (e) gradient magnitude values, bright values indicate edges. (f) *mask* region for morphological reconstruction. (g) background bg_{sF} is subtracted for wing region segmentation, body and non-arena regions are black. (h) binarized wing region with legs L. (i) fly wing region W without legs and with filled holes. (j) original image with segmentation results, body region in red, wing region in blue. (k) cautiously smoothed background bg_{sF}. (l) rigorously smoothed background bg_{SA}. 63
29 Approaches for wing region partitioning. (a) original wing region W. (b) wing region partitioned by voronoi splitting operation. (c) body regions B_1 and B_2 used for competitive flood fill initialization. (d) floodfill-separated wing regions W_1 and W_2, colored in the corresponding body colors in (c). 70
30 Main information flow through the postprocessor, with focus on handling of σ and τ sequences. 76
31 Occlusion sample. (a) original video frame depicting two overlapping flies. (b) extracted body region of the merged bodies in magenta, wing region in yellow. 81
32 Overview of a large variety of t-methods, a few interesting point-based methods and area-based methods are introduced later in this section. 83
33 (a) bwImage of the two regions before Occlusion. (b) extracted characteristic information $\mathfrak{C}$ for method *boc* 86
34 (a) Two pre-occlusion regions $\mathfrak{R}^1$ and $\mathfrak{R}^2$ corresponding to figure **??**a, colored according to known tracking information. (b) partition of $\mathfrak{C}$, according to identifier regions depicted in (a) . . . 87
35 (a) Voronoi diagram for filled fly regions. Each point located in a region will take over the identifier value of the regions voronoi center (b) Voronoi diagram for characteristic information $\mathfrak{C}$ (c) Points of $\mathfrak{C}_{s+1}$ in regions of $V(\mathfrak{C}_s)$ 88
36 (a)-(e): partitions of characteristic information carried through successive frames 226-230 in sample video 1 89

37 (a) partition $\hat{\mathfrak{C}}_a$ of frame 230 carried through from pre-occlusion tracking information (b) partition $\hat{\mathfrak{C}}_{a'}$ of frame 230 out of post-occlusion tracking information (c) sample mapping matrix $M_{b\to a}$ from frame 225 to frame 230, $M_{225\to 230}$, containing votes (i.e. matched identifiers) for odd-odd resp. odd-even mapping 90
38 (a)-(i): boc output for video 1, frames 3077-3087. Although the flies are crossing boc method reports "no cross" here, frame (d) catches reason for this, intrinsically coming with the voronoi carrier 92
39 (a)-(i): boc output for video 1, frames 4592-4600. All identifiers of the male fly are overwritten by female identifiers, resulting in "don't know" for frame 4601. 96
40 (a)-(i): boc output for video 1, frames 3240-3245. Although the flies are crossing boc method reports "no cross" here, frame (c) catches reason for this, intrinsically coming with the voronoi carrier (see also "known weaknesses" below) 97
41 (a) The white line shows the (smoothed) boundary of two occluded objects. The red line shows the boundary of the convex hull, which is build as a polygon of the blue points. (b) depicts the distance between the occluded objects perimeter (depicted in (a) as white line) to the boundary of the convex hull (depicted in (a) as red line), starting at the left-top point of the convex hull. The perimeter and the perimeter of the convex hull are fragmented between their shared points depicted in (b) as blue points), distances are measured between corresponding fragments and are computed by normals to the convex hull fragments. . . . 101
42 (a) sample input partition for the concavity improvement algorithm (b) output partition as algorithm result, where the transition between identifiers is adjusted according to highest peeks in figure **??** (b), as the "most concave" points are likely to be on the fly transition 101
43 Snapshot of the annotationTool, depicting a frame where the usually bigger female fly "turns up" and therefore has smaller *Area* values. The snapshot further visualizes the values of attributes *Area*, *AreaE* and *AreaEC* over time and visualizes that the latter attributes successfully compensate up-turning flies and therefore provide more discriminative size values. 107
44 Classification And Regression Tree (CART) combining output of several t-methods and observable occlusion features. 109

45 Sample problem instance for the SDH-algorithm (a) identified flies over time: A in green and B in orange. Occluded sequences marked by gray boxes. s-values associated with non-occluded sequences in blue, t-values for occluded sequences in brown. The text summarizes the computation of s-values and t-values and color-codes keywords and definition in blue and brown for s- resp. 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. 113
46 Workflow for resolving occlusions. 117
47 Manual overrulement of an occlusion solution. The picture above depicts a correct occlusion assignment, the picture below a manually overruled (but then wrong) identity assignment. 123
48 (left): linear workflow where identities are assigned and data is interpolated accordingly. (right): data is interpolated first and split into two possible worlds. Later, when occlusions are resolved, the worlds are merged and the data flow continues linearly. 128
49 Two sample interpolation for different methods (a) sine-like scenario (b) plateau-like scenario. Blue rings depict the original data points, the green line the pchip interpolation, the red dotted line the spline interpolation. 130
50 Wrong ellipse rotation interpolation. The left and the right picture show the frame before resp. after the occlusion. The picture in the middle one of the interpolated pictures. The interpolated ellipse in the middle picture is obviously rotated in the wrong direction. 131
51 Sequence of interpolated ellipses. The first and the last picture show frames before resp. after the occlusion. The frames in between depict the sequence of interpolated ellipses, overlayed on the original occluded video frame. The female fly (red) rotates in long, the male fly (blue) in short direction. 132
52 Rescued Wing extension during occlusion event. (left): the body regions of the two flies are slightly occluded. (right): the interpolated ellipses fit to the flies body regions and allow to still capture the ongoing Wing extension event. 133
53 Logistic regression in a nutshell. 138
54 (a) everyday-scores of 5 biologists plotted against a ground truth-score of one biologist. (b) everyday-score of the ground truth scoring biologist and the machine learned scores plotted against the ground truth score 140

55 Winning attributes in pictures: (a) *distHeadTail*, the distance between the flies *Head* to the other flies *Tail* (b): *nArea* and *nwArea*, the size of the body resp. the wing area (c) *flyLen*, the *MajorAxislength* of the fitting ellipse and the distance between *Head* and *Tail* (d) *Eccentricity*, a measure for *roundness* that depends on the two focal points of the covering ellipse resp. on the relation between its *MajorAxisLength* and *MinorAxisLength* (e) *minDist* the shortest of the five depicted distances (f) *distance*, the distance between the two flies centers-of-gravity (g) *hrndirectednessR*5, a combined measure of orientation angle and distance (h) *nAngleHeadHead*, the angle between a fly and the other flies *Head*. 143

56 Visualization of following classifier and its underlying attributes. (a) *isOcclusion*, black indicates that the flies body regions overlap each other (b) *Foll*, binary classifiers are visualized by two half bars, the blue half bar is displayed where the attribute is true for the male fly, the red half bar is displayed when the attribute is true for the female fly (c) *movedG*, the blue line depicts values for the male fly, the red line for the female fly (d) *FollB* (e) *distance*, just one black line depicting the distance between the two flies (f) *FollC* (g) *angleHeadTCentroid* (h) *FollD* (i) *moveAngle* (j) *FollE* (k) *distHeadTail* (l) *FollF* (m) *Folled*, the continuity filtered classifier where events shorter than 1 second are eliminated. The screen shot in figure 60 depicts the pictures and all events for this sequence.(. 147

57 Visualization of orientation classifier and its underlying attributes, visualization format as in figure 56. (a) *isOcclusion* (b) *Ori* (c) *angleHeadTCentroid* (d) *OriA* (e) *distance* (f) *OriB* (g) *movedG* (h) *OriC* (i) *distHeadHead* (j) *distHeadTail* (k) *OriD* (l) *Oried* 148

58 Visualization of circling classifier and its underlying attributes, visualization format as in figure 56. (a) *isOcclusion* (b) *Circ* (c) *distance* (d) *CircA* (e) *angleHeadTCentroid* (f) *CircB* (g) *movedG* (h) *CircC* (i) *CircD* (j) *angleDiffG* (k) *CircE* (l) *vaz* (m) *CircF* (n) *Circed* 150

59 Visualization of wing extension classifier and its underlying attributes, visualization format as in figure 56. (a) *isOcclusion* (b) *Wing* (c) *tWTangle* (d) *WingAL* (e) *tWTareaRatio* (f) *WingBL* (g) *WingAR* (h) *WingBR* (i) *WingL* (j) *WingR* (k) *WingBi* (l) *Winged* 152

60 Screenshots of the annotationTool, the green line depicts the position of the displayed picture within the time series. (a) shows picture and summarizes the major events for the time series used in figures 58 and 59. (b) event summary and picture for time series used in figures 56 and 57. 154

61 Color coded ethogram (a) detected events per line, in order: Orientation (blue) fly A, Following (green) fly A, Wing Extension (red) fly A, Circling (magenta) fly A, Orientation fly B, Following fly B, Wing Extension fly B, Circling fly B, Copulation. The last line visualizes the *isOcclusion* attribute. (b) compact visualization of all detected events for fly A. 155
62 Visualization of multiple chambers in a color-coded ethogram. Each line contains a compact view as in figure 61 (b), time of event occurrence is visualized in x-direction. The legend visualizes the color code. 156
63 Behaviour visualization of wild-type behaviour and known mutants. (a) wild-type male behaviour (b) wild-type female behaviour (c) Fru^F male behaviour (d) fru^M female behaviour. 157
64 Screenshot of the webInterface showing the video upload mask and a project summary outlining how many videos are at which status. 160
65 Screenshot of the video list view of the webInterface. The top bar allows to select browse the videos of a selected project, the checkboxes on the left allow to select videos and the dialog on the bottom allows to apply specific operations (like "track", "eject" or "add comment") to the selected videos. The list itself contains the video names, the number of detected chambers, the current status of the video, the date of the last status change and GUI elements to start further processing steps, view processing results or to play the video. 161
66 Screenshot of the per-video report after the preprocessing step, depicting the median background, the pixelwise minimum of the frames that were used to compute that background, the cautiously smoothed background model, the first frame of the video overlayed with the detected arenas, the rigorously smoothed background model and the output text produced by during preprocessor runtime. 162
67 Screenshot of the chamber view of the webInterface, depicting the video summary pictures on the left, per-video acception of rejection pictures, the courtship index summary and GUI elements for further processing, inspection or annotation steps. 163
68 Screenshot of the annotationTool in a non-occluded frame. 166
69 Screenshot of the annotationTool in an occluded frame. 167
70 (a) snapshot of the experimental setup where the spin off variant aimed to analyze. Flies perform in bright chambers and eventually show chaining behavior (b) manually annotations marked in yellow versus an automatic approximation measure marked in red and blue. The sharp rise in the middle happens at the time point when the courtship song was turned on. Both, manual and automatic assessments capture a significant change. 169

71 Preprocessing steps handling food spots (a) snapshot of a sample video. (b) detected food spots visualized in different colors. (c) food spots grouped by detected food color. 170
72 Results of a sample analysis. (a) shows the voronoi regions for each food spot, a food spot surrounding partitioning of the arena. (b) quantifies the percentage of flies per food type over time (c) quantifies the percentage of time that a fly spent directly on a particular food spot, normalized by the total number of flies (d) quantifies the percentage of time that a fly spent in the voronoi region of a particular foodspot, normalized by the total number of flies but *not* normalized by the size of the voronoi area. Subfigures (b), (c) and (d) were computed by Lorenz Pammer. 171
73 Wing Extension Direction Analysis. (a) depicts the female positions when the male fly extended its left wing, (b) depicts the female positions when the male fly extended its right wing. . . . 172
74 Schematic visualization of the biological context for the system's application. (a) biologists are able to modify single or small groups of neurons within the fly brain, the picture shows a schematic fly brain and some genetically targetable neurons (b) flies with manipulated neurons are recorded in videos. (c) flies with manipulated neurons may have differently hard-wired neuronal circuits, the picture shows a schematic neuronal circuit. (d) the behaviour in videos containing such flies with manipulated neurons resp. manipulated circuits are quantified by the system introduced in this thesis, the picture shows a change in ethograms resulting from a manipulated circuit. 175
75 Tracker improvement over time. (a) older version with thresholding and voronoi separated wing areas (b) newest version with gradient corrected thresholding and floodfill separated wing areas. 177

References

[1] Sokolowski, MB Drosophila:Genetics meets behaviour. Nature Rev. Genet. 2, 879–890 (2001)

[2] Demir E. and Dickson B.J. fruitless splicing specifies male courtship behavior in Drosophila. Cell 121, this issue, 785–794 (2005)

[3] Kurtovic A., Widmer A. and Dickson B.J. A single class of olfactory neurons mediates behavioural responses to a *Drosophila* sex pheromone. Nature 446, 542-546 (2007)

[4] Yapici N., Kim Y.J. Ribeiro C. and Dickson B.C. A receptor that mediates the post-mating switch in Drosophila reproductive behaviour. Nature 451, 33-37 (2008)

[5] Petra Stockinger, Duda Kvitsiani, Shay Rotkopf, László Tirián and Barry J. Dickson.. Neurocircuitry that governs male courtship behaviour. Cell 121: 795-807 (2005).

[6] MATLAB® (R2009a, The Mathworks, Natick, MA) documentation

[7] Leo Grady and Eric Schwartz. Anisotropic interpolation on graphs: The combinatorial Dirichlet problem. Technical Report CAS/CNS-TR-03-014, Department of Cognitive and Neural Systems, Boston University, Boston, MA, July 2003. Submitted to IEEE Pattern Analysis and Machine Intelligence.

[8] Leo Grady and Eric L. Schwartz. The Graph Analysis Toolbox: Image processing on arbitrary graphs. Technical Report CAS/CNS-TR-03-021, Department of Cognitive and Neural Systems, Boston University, Boston, MA, August 2003.

[9] P. Hough. Method and Means for Recognizing Complex Patterns. US Patent 3,069,654, 1962

[10] R. O. Duda and P. E. Hart. Use of the Hough Transformation to Detect Lines and Curves in Pictures. Comm. ACM, Vol. 15, pp. 11–15, 1972

[11] D. H. Ballard. Generalizing the Hough Transform to Detect Arbitrary Shapes. Pattern Recognition, Vol.13, No.2, p.111-122, 1981

[12] http://en.wikipedia.org/wiki/File:Hough-example-result-en.png Sept 14th 2010.

[13] Linda G. Shapiro and George C. Stockman (2001): “Computer Vision”, pp 279-325, New Jersey, Prentice-Hall.

[14] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, 1979, pp. 62-66.

[15] Florian Seitner. Robust detection and tracking of objects. Technical Report PRIP-TR-100, University of Technology Vienna, 2005.

[16] Rafael C. Gonzalez, Richard E. Woods, and Steven L. Eddins. Morphological Reconstruction. Matlab Digest Academic Edition, April 2010, online available under http://www.mathworks.com/company/newsletters/edu_digest/2010/apr/morphological-reconstruction-matlab.html

[17] H. W. Kuhn. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, Volume 2, Issue 1-2, p. 83–97, March 1955.

[18] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. The Annals of Mathematical Statistics 38 (2): 325–339, 1967

[19] G. Shafer. A Mathematical Theory of Evidence. Princeton University Press, 1976.

[20] Dobson, A. J., An Introduction to Generalized Linear Models, CRC Press, 1990.

[21] MuCullagh, P., and J. A. Nelder, Generalized Linear Models, 2nd edition, Chapman & Hall, 1990.

[22] Carlos Ribeiro and Barry J. Dickson. Sex Peptide Receptor and Neuronal TOR/S6K Signaling Modulate Nutrient Balancing in Drosophila. Current Biology 20, 1000–1005, June 8, 2010.

[23] T. F. Cootes and C. J. Taylor. Active Shape Models - 'Smart Snakes'. In Proc. British Machine Vision Conference. Springer-Verlag, 1992, pp.266-275

[24] Pierce-Shimomura JT, Dores M, Lockery SR. Analysis of the effects of turning bias on chemotaxis in C. elegans. J Exp Biol. 2005 Dec;208(Pt 24):4727-33.

[25] Cronin CJ, Mendel JE, Mukhtar S, Kim YM, Stirbl RC, Bruck J, Sternberg PW. An automated system for measuring parameters of nematode sinusoidal movement. BMC Genet. 2005 Feb 7;6(1):5.

[26] Feng Z, Cronin CJ, Wittig JH Jr, Sternberg PW, Schafer WR. An imaging system for standardized quantitative analysis of C. elegans behavior. BMC Bioinformatics. 2004 Aug 26;5:115.

[27] Baek JH, Cosman P, Feng Z, Silver J, Schafer WR. Using machine vision to analyze and classify Caenorhabditis elegans behavioral phenotypes quantitatively. J Neurosci Methods. 2002 Jul 30;118(1):9-21.

[28] Martin JR. A portrait of locomotor behaviour in Drosophila determined by a video-tracking paradigm. Behav Processes. 2004 Sep 30;67(2):207-19.

[29] Susanne C. Hoyer, Andreas Eckart, Anthony Herrel, Troy Zars, Susanne A. Fischer, Shannon L. Hardie, Martin Heisenberg. Octopamine in Male Aggression of Drosophila. Current Biology - 12 February 2008 (Vol. 18, Issue 3, pp. 159-167).

[30] Heiko Dankert, Liming Wang, Eric D Hoopfer, David J Anderson and Pietro Perona. Automated monitoring and analysis of social behavior in Drosophila. Nature Methods 6, 297 - 303 (2009).

[31] Kristin Branson, Alice A. Robie, John Bender, Pietro Perona and Michael H. Dickinson. High-throughput ethomics in large groups of Drosophila. Nature Methods 6, 451 - 457 (2009)

[32] Dhruv Grover, John Tower and Simon Tavare. O fly, where art thou? J. R. Soc. Interface (2008) 5, 1181–1191.

Acknowledgements

First of all I want to thank Barry Dickson for giving me the unique opportunity to work as the only computer scientist in his biology lab.

I've learned a lot in these 5 years at the I.M.P.

As my supervisor Barry guided me through different stages of the system development, I really appreciate that he invested so much time and enthusiasm into this project. Barry extensively analyzed data and came up with valuable input and ideas, especially for the postprocessing part.

Secondly, thanks to Laszlo Tirian, a Postdoc in Barry's lab and my main biology collaboration partner. Laszlo was my test pilot in every stage and provided my with an extensive amount of video data and manual annotations.

Further, thanks to all of my biology collaboration partners.

Regarding this thesis, particularly thanks

- to Laszlo Tirian, Christopher Masser, Sabrina Jörchel, Krystyna Keleman, Jai Yu, Anne von Philipsborn, Simone Latkolik, Amina Kurtovic for recording courtship videos that were processed during different stages of the software development,
- to Laszlo Tirian, Ebru Demir, Alex Widmer, Amina Kurtovic, Krystyna Keleman for manually annotating courtship behaviour,
- to Laszlo Tirian and Sebastian Krüttner for extensively annotating courtship behaviour,
- to Laszlo Tirian, Sabrina Jörchel and Krystyna Keleman for manually inspecting occlusions
- to Laszlo Tirian and Sabrina Jörchel for testing the webInterface and the annotationTool
- and to Laszlo Tirian for teaching others how to operate these tools.

The figures of this thesis were mainly computed from videos given by Laszlo Tirian and Christopher Masser, a few pictures were computed from videos given by Krystyna Keleman and Jai Yu.

Further thanks to Kai Feng and Carlos Ribeiro for the collaborations that lead to the spin-offs in section 6.3.1 and 6.3.2. When Kai asked for a multi-fly tracker in a different setup an early branch of the code was made and the chaining detector in section 6.3.1 was developed. Kai later developed another postprocessor module for his own purposes without further support.

The second spin-off was applied in Carlos' setup (see section 6.3.2), where a summer student implemented a postprocessor module for analyzing foraging behaviour based on a later branch of the code. The preprocessor adaption and voronoi region computation was computed by myself, thanks to Lorenz Pammer for implementing the whole analysis part and for providing me with the the pictures shown in figure 72 (b)-(d).

Special thanks to Thomas Micheler for the web interface and thanks to Wolfgang Lugmayr for curation of the computer cluster, this front end and the computing power behind it allowed to bulk-process hundreds of videos with a few mouse clicks. Especially Thomas invested a lot of time for implementing and adapting dialogs, for integration tests with new major releases of the software and for sharing his web server with us, thanks also to the VDRC for funding this.

Further, special thanks to Alex Gabler who joined the project as an external programmer for 2-3 man months of pressurous work, Alex performed numerous occlusion analysis tasks and contributed to the quality control part of the preprocessor.

Also thanks to Christian Machacek who took over the project for future maintenance and currently optimizes the code for speed. Christian further develops a GUI that allows to organize and process videos on a standard laptop. To my best knowledge Christian implemented another spin-off for analyzing walking behaviour.

Thanks to Alex Keene, to Frank Schnorrer and to Eleftheria Vrontou for small collaborations in side projects. With Alex a simple pulse song analysis for the Drosophila courtship song was implemented, for Frank a simple heartbeat analyzer for microscopy videos was implemented and with Eleftheria the Wuerzburg tracker from [29] was set up for her aggression assay.

Very special thanks to Laszlo Tirian for his patience and dedication to the project.

Very special thanks also to Prof. Wilfried Grossmann, the university host of my PhD, for his great interest in the project, for answering numerous questions of mine and for his great support during the final phase of the project.

Special thanks to Agata Baranowska and Dietlind Frey for proof-reading this thesis.

Very special thanks to Agata Baranowska for her great support and for sharing this experience with me. Agata was not utterly amused when she heard about my PhD plans but still supported me in all regards, amongst others with financial resources and sanity.

Scientific CV

EDUCATION:

- Since 2006: PhD (Computational Biology and Scientific Computing), University of Vienna and PhD program at the Research Institute for Molecular Pathology (I.M.P.), Dickson Lab.
  Dissertation: "Computational Analysis of Drosophila Courtship Behaviour",
  Advisors: Dr. Barry Dickson, I.M.P and Prof. Wilfried Grossmann, Institute for Scientific Computing
- 2008: M.S. with Honors (Economics and Computer Science), University of Vienna
  Specialisation: Distributed Systems
- 2006: M.S. with Honors (Computational Intelligence), Technical University of Vienna
  Thesis: "Bionic Approaches for the Semantic Web",
  Advisor Prof. Eva Kuehn, Institute for Computer Languages
  Specialisation: Theoretical Computer Science and Logic
- 2003: Visiting Student at the University of Edinburgh, Laboratory for Foundations of Computer Science

  Proof of concept for a heterogeneous, distributed database system
- 2002: B.S. (Software and Information Engineering), Technical University of Vienna
  Specialisation: Software Engineering
- 1994: A-Level with Honors (Secondary College of Engineering), HTL Vienna XXII
  Specialisation: Electronic Data Processing and Organisation

PROFESSIONAL EXPERIENCE (in scientific environment): .

- 9/2006 - 9/2010: Research Institute for Molecular Pathology, PhD Student in Dickson Lab
  PhD Project "Computational Analysis of Drosophila Courtship Behaviour",
  Design and development of a fly tracker – software that converts videos into a time series and downstream analysis for such time series, the software enables detailed video analysis and supports new biological insights.
- 6/2005 - 9/2006: Institute for Molecular Biotechnology Austria, Software Architect in Dickson Lab, later Vienna Drosophila Research Center
  Design and development of fail-safe software for the RNAi fly library, processing high data amounts in near- time.

- 3/2005 - 7/2005: Technical University, Institute for Computer Languages. Tutor for the lecture "Distributed Programming with Space Based Computing Middleware".
- 4/2003 – 7/2003: Technical University of Vienna, Institute for Communication Networks, Research Associate
  Feasibility study of the combination of authentication protocols, development of a wireless-LAN application