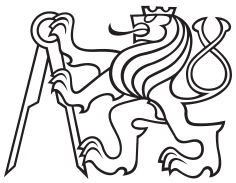
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Tracking, Learning and Detection of Multiple Objects in Video Sequences

Filip Naiser

Supervisor: prof. Ing. Jiří Matas, Ph.D.
January 2017

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Filip N a i s e r

Study programme: Open Informatics

Specialisation: Computer Vision and Image Processing

Title of Diploma Thesis: Tracking, Learning and Detection of Multiple Objects in Video Sequences

Guidelines:

1. Get familiar with multiple object-tracking (MOT).
2. Review the state-of-the-art, focussing on MOT methods that include tracking, learning and detection.
3. Define MOT as cost minimisation, consider formulation as graph labelling.
4. Learn cost functions parameters using semi-supervised machine learning methods.
5. Design and learn a classifier for object re-detection.
6. Evaluate the method on realistic sequences. Include comparison with published algorithms.

Bibliography/Sources:

[1] D. A. Forsyth, J. Ponce: Computer Vision: A Modern Approach. Prentice Hall, 2011.
[2] R. Szeliski: Computer Vision: Algorithms and Applications. Springer, 2010.
[3] Murphy, Kevin P.: Machine learning: a probabilistic perspective. MIT press, 2012
[4] Chapelle, Olivier, Bernhard Schölkopf, and Alexander Zien: "Semi-Supervised Learning. Adaptive Computation and Machine Learning series." (2006)

Diploma Thesis Supervisor: prof. Ing. Jiří Matas, Ph.D.

Valid until: the end of the winter semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, May 25, 2016

Acknowledgements

I would like to express my gratitude to my supervisor prof. Ing. Jiří Matas, Ph.D., for his valuable advice, guidance, patience and willingness during my study.

I would also like to give my thanks to Barbara Casillas-Perez from the Cremer Group at the Institute of Science and Technology Austria for being helpful in problem discussions and software debugging, and for her patience in the beginning when I had trouble overcoming the language barrier.

Furthermore, I would like to thank Czech Technical University students Dita Hollmannová and Šimon Mandlík for their effort invested in the implementation of related tools, mostly during their summertime jobs.

My thanks also go to my family for their support and patience, especially during exam times.

Lastly, I would like to thank Cremer Group from the Institute of Science and Technology Austria for hosting me from February 2015 till June 2015.

The research was supported by CTU student grant SGS15/155/OHK3/2T/13.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within accordance with the methodical instructions for observing ethical principles in the preparation of university theses.

Prague, 9th January, 2017

Abstract

The domain of Multi-Object Tracking encompasses various interesting problems including animal surveillance in biology experiments. Automated or semi-automated evaluation of experiments has a tremendous impact in biology. In this thesis, we are introducing a method for tracking of multiple interacting objects in laboratory conditions. The difficulties of identity maintenance arising from object interactions and occlusions are solved by identity re-detection. We have enhanced current state-of-the-art classifier for objects hardly distinguishable by a naked eye, and we have shown that it improves the performance for per region classification by 4.5%–18% on video sequences with ants, zebrafish, and bugs. The method implementation is accompanied with a graphical user interface. It includes tools for manual annotation of difficult object interactions that hint the tracker and also means for final correction of tracking output.

Keywords: Multi-Object Tracking, animals, insects, learning, detection, identity maintanence, Random Forest Classifier

Supervisor: prof. Ing. Jiří Matas, Ph.D.
Karlovo náměstí 13,
121 35 Praha 2

Abstrakt

Sledování většího počtu objektů ve video sekvencích v sobě zahrnuje mnoho nejrůznějších podproblémů, včetně sledování zvířat během biologických experimentů. Jejich automatické, či poloautomatické vyhodnocení, má pro biology obrovský význam. V této tezi představujeme metodu pro sledování většího počtu objektů v laboratorních podmínkách. Obtíže způsobené možnou ztrátou identity během interakcí, kdy dochází k nepřehledným situacím, jsou řešeny pomocí rozpoznávání identity jedince. Rozšířili jsme aktuálně nejlepší metodu pro klasifikaci okem těžko rozeznatelných objektů a na sekvencích zachycujících mravence, ryby a svinky jsme ukázali, že tímto způsobem lze dosáhnout zlepšení o 4,5-18%. Metodu jsme implementovali a doplnili o uživatelské rozhraní pro zobrazení a editaci výsledků, umožňující interakci a doplnění informací vprůběhu výpočtů.

Klíčová slova: Počítačové vidění, sledování několika objektů, zvířata, hmyz, učení se, detekce, udržování identity, Les randomizovaných rozhodovacích stromů

Překlad názvu: Sledování, učení se a rozpoznávání objektů ve videosekvencích

Contents

Nomenclature 1

1 Introduction 3

1.1 FERDA, a Semi-Automatic Labeling System 4

2 Related work 7

3 Problem formulation 11

4 Approach 15

4.1 Segmentation Step 15

4.1.1 Maximally Stable Extremal Regions (MSER) 16

4.1.2 Random Forest Pixel Classifier 16

4.2 Graph Modifications 16

4.2.1 Isolation Forest 17

4.2.2 Training Data Acquisition . . . 17

4.2.3 Motion Model 18

4.2.4 Region Alikeness Model 18

4.2.5 Full ID-Set Transfer Decision Function 19

4.3 Region Classification 19

4.4 ID assignment Constraints 20

4.5 ID assignment Detection 21

4.6 Expert Annotation 22

5 Parameters Tuning 23

5.1 Decision Threshold - ζ 23

5.2 Number of labeled samples for *Region* Classification training 23

5.3 Random Forest Classifier (RFC) - Parameter Tuning 26

5.3.1 Features 26

5.3.2 Split Criterion Function 28

5.3.3 Max Depth of a Decision Tree 29

5.3.4 Min Samples In a Leaf 29

5.3.5 Max Features 30

5.3.6 Number of Decision Trees . . . 31

5.4 RFC Overall Setting 32

6 Dataset 33

6.1 Camera3 33

6.2 Cam1 35

6.3 Zebrafish 35

6.4 Sowbug3 36

7 Experiments 39

7.1 ID-Classifier Comparison 39

7.2 FERDA's Overall Performance . 39

7.2.1 With No Expert Annotations 40

7.2.2 With Expert Annotations . . . 41

7.3 Results Discussion 44

8 Implementation Details 45

8.1 Video Compression 46

8.2 Graph Viewer 46

8.3 Results Viewer 47

8.4 RFC Segmentation Tool 48

9 Conclusion 51

9.1 Future Work 51

Bibliography 53

A RFC Tuning Results 57

A.0.1 Max Depth of a Decision Tree 57

A.1 Min Samples In a Leaf 58

A.1.1 Max Features 59

A.1.2 Number of Decision Trees . . 62

B Tracker Comparison Diagrams 65

Figures

1.1 (The correct answer is the same for all: 1E, 2D, 3C, 4B, 5A) 6

2.1 An example of rapid movement from Ants-1 dataset. This dataset has been acquired and kindly made available by Barbara Casillas-Perez, Cremer Group, Institute of Science and Technology Austria. The framerate is 15fps. 8

4.1 An example of a sequence where multiple size objects are present. In this sequence, the segmentation is very challenging. This image has been acquired and kindly made available by Barbara Casillas-Perez, Cremer Group, Institute of Science and Technology Austria. 16
4.2 Euclidean distance of centroids vs perpendicular distance of *region* circumscribed rectangles. 17
4.3 Segmentation class examples from Cam1 dataset. 20

5.1 An impact of a number of **k**-diversely labeled samples and a feature sets on 1-NN *region* classifier accuracy. Feature set f1 = {region area, major/minor axes length, contour length}, f2={f1, *region* min. intensity}, f3 = {f2, *region* max. intensity}, f4 = {f3, *region* margin} 24
5.2 An impact of a number of **k**-diversely labeled samples and a feature sets on 1-NN *region* classifier accuracy. Feature set f1 = {region area, major/minor axes length, contour length}, f2={f1, *region* min. intensity}, f3 = {f2, *region* max. intensity}, f4 = {f3, *region* margin} 25
5.3 An examples of I-co-occurrence matrices for 2 different object identities 27
5.4 An examples of I-co-occurrence matrices for 2 different object identities 28

6.1 Camera3 dataset frame samples. 34
6.2 An interaction example sequence in dataset Ants-3. 34
6.3 Cam1 dataset frame samples. . . . 35
6.4 Zebrafish dataset frame samples. 36
6.5 Sowbug3 dataset frame samples. 36

7.1 ID detection results on Sowbug-3, Ants-1, Ants-3, Zebrafish-1 datasets. Y axis labels identities, X axis labels frames. Top row - idTracker (**without** interpolation), middle (thinner) row - ground truth, bottom row - FERDA (**with** expert help during initialisation). Gray - ID unassigned. 42
7.2 ID detection results on Sowbug-3, Ants-1, Ants-3, Zebrafish-1 datasets. Y axis labels identities, X axis labels frames. Top row - *idTracker* (**with** interpolation), middle (thinner) row - ground truth, bottom row - FERDA (with expert help during initialisation). Gray - ID unassigned. 43

8.1 A screenshot from FERDA-graph-viewer tool implemented by Šimon Mandlík depicting a progress of tracklet creation. The edge color visualizes the likelihood of full ID-Set transfer starting with red (not likely) continues through yellow to green (very likely).) 47

8.2 A screenshot from FERDA-results-viewer tool. In the middle part, there is a video player with results showed. P and N ID-Sets are visualized by stripes with squares. Where color describes an ID. When it is crossed, it is in N set, when enhanced it is in P set. When segmentation color is dotted it does not represent object ID but tracklet id. Belov is the video player and visualization control panel. In the left part, controls for GT creation, other system's results from visualisation and comparison and debugging tool are situated. 48
8.3 Example of RFC, per pixel classification outputs. Squares show user input marks, green represents foreground and purple represents background - legs and antennas are to be ignored in this case. Light green is the output probability map computed by RFC. Orange color shows resulting MSER contours, computed on probability map. 49

B.1 ID detection results on Sowbug-3, Ants-1, Ants-3, Zebrafish-1 datasets. Y axis labels identities, X axis labels frames. Top row - *idTracker* (**with** interpolation), middle (thinner) row - ground truth, bottom row - FERDA **(without ID-Consistency Rules and without expert annotations)**. Gray - ID unassigned. 65
B.2 ID detection results on Sowbug-3, Ants-1, Ants-3, Zebrafish-1 datasets. Y axis labels identities, X axis labels frames. Top row - idTracker (**without** interpolation), middle (thinner) row - ground truth, bottom row - FERDA **(without ID-Consistency Rules and without expert annotations)**. Gray - ID unassigned. 66
B.3 ID detection results on Sowbug-3, Ants-1, Ants-3, Zebrafish-1 datasets. Y axis labels identities, X axis labels frames. Top row - *idTracker* (**with** interpolation), middle (thinner) row - ground truth, bottom row - FERDA **(without expert annotations)**. Gray - ID unassigned. 67
B.4 ID detection results on Sowbug-3, Ants-1, Ants-3, Zebrafish-1 datasets. Y axis labels identities, X axis labels frames. Top row - idTracker (**without** interpolation), middle (thinner) row - ground truth, bottom row - FERDA **(without expert annotations)**. Gray - ID unassigned. 68

Tables

5.1 Random Forest Classifier performance default values set to: **criterion** = entropy; **max-f** = 50%; **min-leaf** = 1; **n-trees** = 10; **max-d** = unlimited. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.. All following results are compared with this table. Better performance is highlighted in green. 26
5.2 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **criterion** = gini. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 29
5.3 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-d** = 10. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 29
5.4 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **min-leaf** = 3. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 30
5.5 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 40. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 30
5.6 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 60. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 31
5.7 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 100. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 31
5.8 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 200. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 32
5.9 Tuned RFC performance difference compared to default settings. 32
5.10 **The overall comparison of tuned RFC on different feature sets.** The last row shows tuned RFC performance, when the combination of all features was used. 32

6.1 Dataset overview. In column GT is the frame interval covered by ground truth. In column #px is the average object pixel resolution. 33

7.1 Overall comparison of different classifier's performance in ID-Assignment task. 39
7.2 Results of tracking system performance on four datasets described in chapter 6. The *idTracker* column is a regular idTracker output, *idTracker I* is an output with provided interpolation. 40
7.3 Results of tracking system performance on four datasets described in chapter 6. The *idTracker* column is a regular idTracker output, *idTracker I* is an output with provided interpolation. 41

7.4 Results of tracking system performance on four datasets described in chapter 6. The *idTracker* column is a regular idTracker output, *idTracker I* is an output with provided interpolation. FERDA was provided with expert annotations. There were 1 for Sowbug-3 sequence, 48 for Ants-1, 30 for Ants-3 and 6 for Zebrafish. 43

8.1 A statistics of *FERDA* project showing the comprehensiveness of this work. It descirbes the number of lines except for the files column, describing number of files in given languge. 45

A.1 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-d** = 5. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 57

A.2 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-d** = 10. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 57

A.3 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-d** = 15. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 58

A.4 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **min-leaf** = 2 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 58

A.5 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **min-leaf** = 3. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 58

A.6 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **min-leaf** = 5 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 59

A.7 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 10 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 59

A.8 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 20 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 59

A.9 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 30 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 60

A.10 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 40. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 60

A.11 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 60. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 60
A.12 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 70 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 61
A.13 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 80 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 61
A.14 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 100 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 61
A.15 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = $\sqrt{f}$, where f is the numbef of features . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 62
A.16 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 20 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 62
A.17 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 30 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 62
A.18 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 40 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 63
A.19 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 50 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 63
A.20 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 75 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 63
A.21 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 100. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 64
A.22 The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 200. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes. 64

Nomenclature

Terminology

pixel	Pair of coordinates (i, j) in an image.
region	Set of image pixels forming a connected component.
ID	Object identity, a unique number denoting a specific individual throughout the video sequence.
Full-ID-Set	Set of all IDs present in a video sequence.
ID-Set	Subset of Full-ID-Set (empty set included).
tracklet	A sequence of regions with the same ID-Set.
ICR	ID-Consistency Rules, the set of rules which must hold for ID-Sets of tracklets.
CToIS	Complete Transfer of ID-Set.
expert annotation	Input provided by the user in reaction to a run-time prompt.
HCI	Human-Computer Interface.
object interaction	Event in which two or more objects occlude each other, resulting in loss of identity due to segmentation failure.
RFC	Random Forest Classifier - a machine learning technique.
k-NN	k Nearest Neigbors - a machine learning technique.
NN	Denotes 1-NN.
GT	Ground Truth, a notion of data which is known to be correct and thus used for experiment evaluation.

Region Related

$\mathcal{A}(\mathbf{r})$	Area of region $\mathbf{r}$ in pixels.
$\mathbf{C}(\mathbf{r})$	Centroid of region $\mathbf{r}$.
$\lambda_1(\mathbf{r})$	Length of the major axis of the ellipsis having the same central moments μ_{20} and μ_{02} as region $\mathbf{r}$.
$\lambda_2(\mathbf{r})$	Length of the minor axis of the ellipsis having the same central moments μ_{20} and μ_{02} as region $\mathbf{r}$.
$I_{\mathbf{min}}(\mathbf{r})$	Minimum grayscale intensity of a pixel in region $\mathbf{r}$.
$I_{\mathbf{max}}(\mathbf{r})$	Maximum grayscale intensity of a pixel in region $\mathbf{r}$.

Functions

$f_T(\mathbf{r}_1, \mathbf{r}_2)$	Decision function determining whether CToIS from $\mathbf{r}_1$ to $\mathbf{r}_2$ should take place, based on motion and region alikeness likelihood.
$f_m(\mathbf{r}_1, \mathbf{r}_2)$	Likelihood of full ID-Set transfer from $\mathbf{r}_1$ to $\mathbf{r}_2$ based on motion.
$f_a(\mathbf{r}_1, \mathbf{r}_2)$	Likelihood of full ID-Set transfer from $\mathbf{r}_1$ to $\mathbf{r}_2$ based on region alikeness.

Chapter 1

Introduction

General Multi-Object Tracking (MOT) is a difficult task due to the diverse domain of objects (e.g. cars, animals, humans, cells, insects) moving in various environments (indoor, outdoor, streets, wilderness, sea), and due to the wide range of problems encountered (e.g. occlusion, object interactions, object disappearance, camouflage, appearance metamorphosis). Unsurprisingly, MOT is in practice divided into subdomains specializing on smaller problems, using domain constraints to their advantage. Having considered the availability of relevant data, challenging problems (e.g. see Figure 1.1), and a high demand for animal tracking tools, we have chosen a specific MOT subdomain - animal surveillance in biology experiments.

The increased demand for computer vision methods for animal surveillance in biology experiments is evident due to the following facts:

A There are numerous frequently-cited publications about tools for analysis of animal behavior. Some of them will be mentioned here, for a detailed description of current approaches and methods see chapter 2. A program called Ctrax[1] published by Branson et. al. in 2009 in the paper *"High-throughput ethomics in large groups of Drosophila"*[2] with more than 300 citations, *"JAABA: interactive machine learning for automatic annotation of animal behavior."*[3] published in 2013 in Nature methods by Kabra et al. with more than 80 citations, or *"idTracker: tracking individuals in a group by automatic identification of unmarked animals"*[4] also published in Nature methods in 2014 by Escudero et. al. with more than 70 citations so far.

B There is a high number of research publications where such methods were used to process and examine experimental data.

C On July 2014, Dell et al. included in their review "Automated image-based tracking and its application in ecology" a "Call to developers"[5] summarizing the key features of the ideal system for the task of animal monitoring in the field of ecology.

D There is much activity in the form of questions, bug reports, and enhancement suggestions on discussion forums dedicated to the above-mentioned tools.

Most of the applications mentioned in the point B share the following common characteristics:

- there are multiple objects in the scene
- the objects are of the same or similar species (e.g. ants, mice, fish, flies)
- the objects are moving in an artificial, controlled environment (e.g. petri dish with an almost uniform background; empty, shallow aquarium; simple maze)
- there are constant lighting conditions
- there is a single static camera, placed above the scene

In a situation that warrants all of the above, the objects are

- distinguishable from background,
- almost scale-invariant,
- possibly mutually indistinguishable with the naked eye.

Additionally, the number of objects stays the same throughout a video sequence (except for special cases like cell divisions or very long-term experiments during which new objects are born). All these facts are important to keep in mind because they are what makes the task different from other *Multi-Object Tracking (MOT)* disciplines (e.g. crowd surveillance, vehicles surveillance, sports tracking).

While the existing software satisfies some of the fields requirements (in table Call for developers, Dell 2014), there is still none that satisfies them all. Furthermore, the projects of the Cremer Group at IST Austria and, in particular, our collaboration partner Barbara Casillas, entail most of these requirements and some additional ones. The projects study the behaviors of single ants and groups of ants in the presence of disease carrying individuals; it is crucial to preserve the identity of individuals, especially during interactions. Long term observation of complete ant colonies requires software flexibility to support changing background and differently sized individuals. It is to address both the general requirements of the field and those of our collaborators that we have developed *FERDA*.

In this thesis, we introduce a novel approach combining multiple object tracking with semi-automatic identity classifier learning used for re-detection once the tracking fails.

1.1 FERDA, a Semi-Automatic Labeling System

FERDA is a system for semi-automatic video sequence annotation. It focuses mainly on multiple, nearly indistinguishable objects (e.g. ants, fish, flies, mice). Objects are moving in nearly 2D (planar) space (e.g. Petri dish,

shallow aquarium, terrarium) which guarantees approximately constant scale. The input is a video sequence, number of objects **K** and user provided expert annotations. Every image from a sequence is reduced into a set of regions. A region is labeled with a subset of **K** identities. The problem is designed as a search of **K**-(almost)-disjoint paths in a graph where the vertices are regions in consecutive frames and the eges represent the probability of assignmnt between them.

The word almost means that paths can overlap during object's interactions and once the interaction ends object's identities are reassigned using identity classifier (to get a feel of *ID assignment* difficulty, see Figure 1.1 and try *ID assignment* on your own). The *ID assignment* are assessed with certainty. When the certainty drops below a certain threshold, the user is asked to provide an annotation (e.g. "this region represents an object A" or, if the *ID assignments* are not possible, due to bad visibility of identifying features, "this object in frame **f** is the same as the object in $\mathbf{f} + d$", where d is typicaly small number of frames representing e.g. 10 seconds of video). The goal is to return k-disjoint paths with a minimum cost while the path starts on the first and ends on the last frame. The cost is expressed as a sum of costs for expert's annotations and the costs for mistakes.

ID assignment challenge. Try to pair animals in the first row with those in the second. Correct answer is shown in the caption.

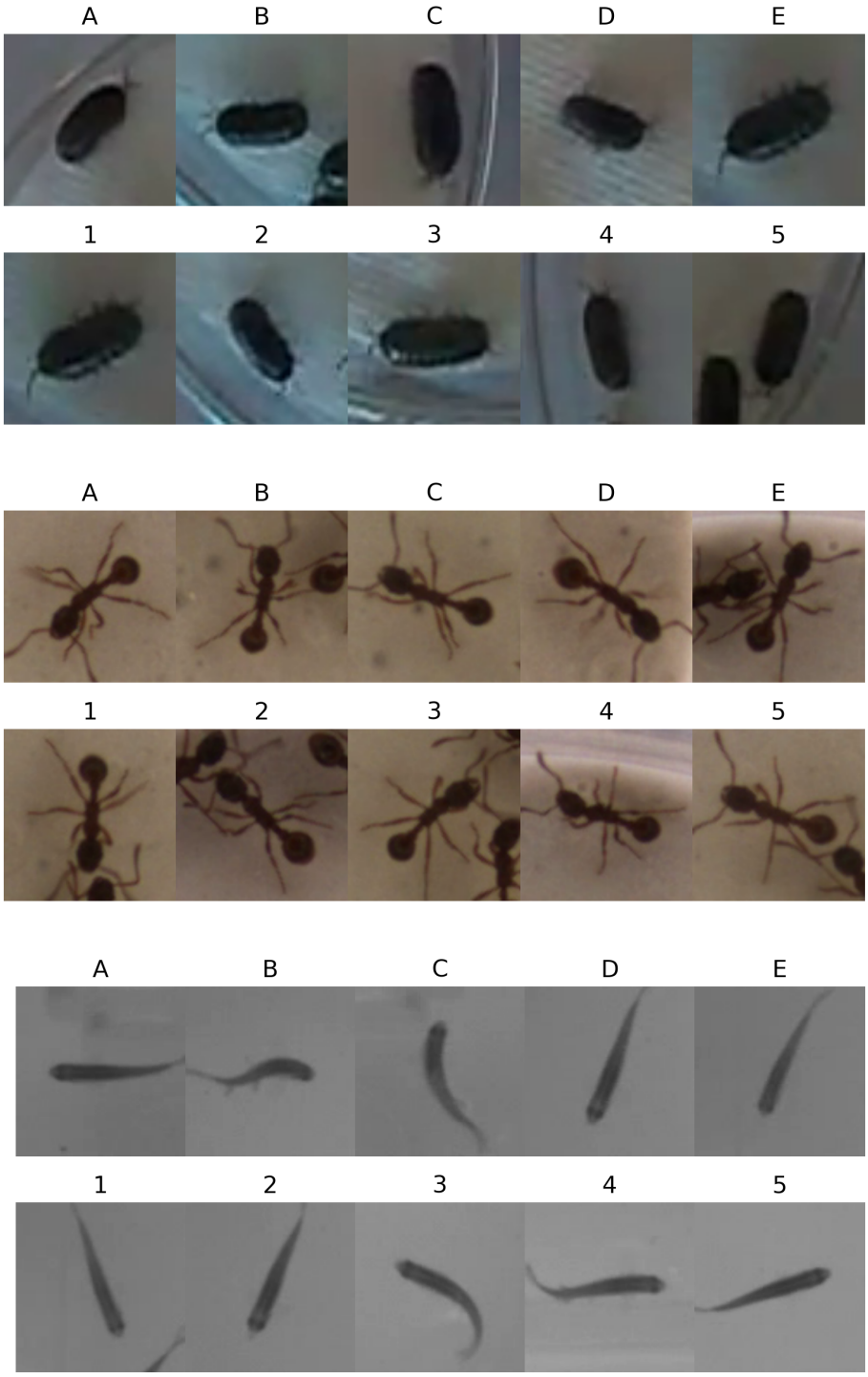


Figure 1.1: (The correct answer is the same for all: 1E, 2D, 3C, 4B, 5A)

Chapter 2

Related work

There are several approaches to animal tracking, which can be categorized according to various criteria. One is whether tagging of individuals is possible (e. g. is object large enough for a tag attachment, does a tag affect the behavior of observed animals). The method of marking is addressed by many researchers. We will mention those related to tracking of smaller objects (e.g. insects, rodents, fish). Tagging using *RFID* is described by Henry et al. [6]or by Schneider et al. [7]. O'Neal et al. [8] and Psychoudakis et al.[9] delineated methods using radio receivers and radars.

We are trying to show the diversity of possible tasks and where our method is situated rather than defining the whole categorisation. Dell et al. [5] have already done very good job in categorization. They have summarized the needs of animal behavior analysts and they have reviewed the current state of the art in animal tracking.

When *RFID* tagging is not possible or desirable, then computer vision based methods should be employed. This category is the one we will discuss next. There are several ways of method categorisation. Based on:

1. the dimensionality of space where we do the tracking (i.e. 3D or 2D).
2. the habitat (i.e. in the laboratory or in the natural environment).
3. the number of tracked objects (e.g. one, a few (2-10) or many (10+)).
4. the information it provides (e. g. the position, identity, behavior labels).
5. the object variety (e.g. objects of the same class like ant workers, or multiple (e.g. predator and prey, workers and the queen).

From now on we will only focus on methods for multiple nearly indistinguishable objects tracking in 2D (planar) space with single camera setting, providing the position and maintaining object identity.

The behavior labeling could be done using specialised software like *JAABA* - an interactive machine learning tool for automatic annotation of animal behavior published in 2012 by Kabra et al. [3]. It is an interactive system, working on top of the provided trajectories, which lets the user to encode their intuition about behavior by annotating a small set of video frames. These manual labels are converted into classifiers that can automatically

annotate behaviors in screen-scale data sets. This system can create a variety of individual and social behavior classifiers for different organisms, including mice and adult and larval *Drosophila*.

The last classification we do is based on the method's ability to reassign an ID after it has been lost (e.g. segmentation failed; it is occluded by the environment or most commonly - lost during multiple animal interactions). Peréz et al. have shown in their paper [4] that when a tracker is not equipped with a method for ID re-detection once the object is lost, there is a high risk of identity swap which will propagate through the whole video sequence. This usually leads to the unusability of identity information and the tracker can be used only for global statistics (e.g. movement speed, occurrence heat maps, the number of interactions).

Ctrax method published in 2009 in Nature Methods by Branson et al. [2] is freely available [1] widely used for *Drosophila* tracking. The object region is detected by thresholding a gray scale image. Centroids of identified individuals are connected through all video frames to yield trajectories. When two animals cross or overlap and oversegmentation occurs, there is a cross solver which fits ellipses and divides such region. Even if the primary focus is on *Drosophila*, it is possible to use it for objects with a shape close to the ellipses. It requires a constant and uncluttered background. Individuals must have similar size.

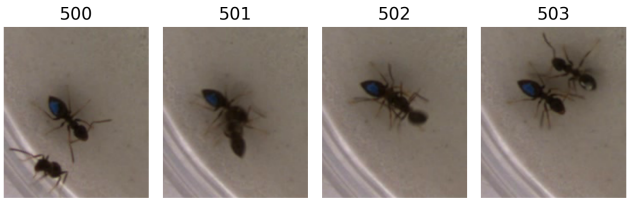


Figure 2.1: An example of rapid movement from Ants-1 dataset. This dataset has been acquired and kindly made available by Barbara Casillas-Perez, Cremer Group, Institute of Science and Technology Austria. The framerate is 15fps.

In 2013 Kimura et al. published a paper [10] on multiple honey bees tracking. It is available in the application called K-Track [11]. It works similarly as *Ctrax*. Authors claim better performance on honey bees data sequences. Just to mention other specialized tools, the Multi-Worm Tracker published by Swierczek et al. [12] is a real-time system which simultaneously quantifies the behavior of dozens of Caenorhabditis Elegans in a Petri dish.

The above mentioned methods don't have ID reassignment capability thus we do not suggest their usage for tasks where an ID maintenance is crucial. An enormous progress in handling the issue of ID maintenance brought a paper called *idTracker* tracking individuals in a group by automatic identification of unmarked animals", published in June 2014 by Peréz et al. [4] in Nature Methods. They have proposed a method capable of individual distinction even in cases where a human fails.

The objects are detected by thresholding of grayscale images (the uniform and uncluttered background is assumed). Then the regions in consecutive

frames are grouped into fragments (tracklets). Two regions belong to the same fragment if they overlap with each other and do not overlap with any other blob of the two frames. This might raise errors when a framerate is low, or the object speed is high or a troublesome combination of both occurs (This approach will fail in a sequence with ant jump shown in Figure 2.1). Once the fragments are constructed, complete sets of fragments are sought. Fragments from the same complete set must have at least one frame in common. The complete set must have exactly **K** fragments, where **K** is a number of observed objects provided during initialization.

A region is described with intensity and contrast co-occurrence matrices. A set of ID examples is established by finding permutations within complete sets of fragments. The regions are classified using nearest neighbor classifier. The metric are two numbers - the mean of per element difference of co-occurrence matrices (one for intensity and one for contrast). The fragment id is assigned based on region classification. If the probability of assignment is too low, it is left unassigned. The method was implemented and is freely available [13].

The author of this thesis has published a bachelor thesis on animal tracking, "Detection, Description and Tracking of Ants in Video Sequences"[14]. The software described therein belongs to the same category as *Ctrax* as it is only a tracker with a heuristics-based interaction solver, but, inherently, once two IDs are swapped, the error propagates through the rest of the video. The problem was modelled as a bipartite graph, where nodes were regions and edges were described by a cost based on change in rotation and the distance from a location generated by movement prediction. A frame to frame assignment problem was solved by maximum weighted matching. The work in this thesis is substantially different from that in bachelor thesis.

Chapter 3

Problem formulation

The input of our method is a video sequence which is an ordered set of images ($\mathcal{S} = \{\mathbf{I}_0, \mathbf{I}_1, \ldots, \mathbf{I}_\mathbf{F}\}$), the number of objects $\mathbf{K}$ and the set of expert annotations $\mathcal{T}$. Before the annotation process starts, the entire video sequence and the number of objects $\mathbf{K}$ must be available. The expert annotation is not a standard input in the way of being provided in the beginning rather then it is an interaction with software during a runtime. For now, we get along with an expert modeled as a function $\mathcal{E} : \mathcal{Q} \to \mathcal{A}$, where $\mathcal{Q}$ is a set of questions and $\mathcal{A}$ is a set of answers. The detailed description of questions and answers set is described in section 4.6.

The output is $\mathbf{K}$-paths, one path for each identity from identity set Γ ($|\Gamma| = \mathbf{K}$). A path is a sequence of *regions*. A *region* is a set of image pixels forming a connected component. The minimum *region* description is a pair (x, y) - the position coordinates. Regions might be supplemented with more detailed descriptions (e.g. "in interaction", "missing", an orientation, area, set of all *region* points, etc.).

We simplify the problem of locating k objects in the images by reducing the search space into a set of *regions* assuming that objects are either inside a *region* or not detected. For all images $\mathbf{I} \in \mathcal{S}$ a segmentation is performed. This way sets of *regions* $\mathcal{L}_\mathbf{f} = \{\mathcal{L}^0_\mathbf{f}, \mathcal{L}^1_\mathbf{f}, \ldots\}$ in all frames $\mathbf{f}$ are constructed.

The temporal and spatial relationship of *regions* is captured in an oriented, complete, bipartite graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$.

$$\mathbf{V} = \bigcup_{\mathbf{f}=0}^{F-1} \bigcup_{i=0}^{|\mathcal{L}_\mathbf{f}|-1} \mathcal{L}^i_\mathbf{f},$$

where F is the total number of frames.

An oriented edge is created for all frame consecutive pairs.

$$\mathbf{E} = \left\{ \forall \mathbf{r}_i \in \mathcal{L}_\mathbf{f}, \forall \mathbf{r}_j \in \mathcal{L}_{\mathbf{f}+1} : (\mathbf{r}_i, \mathbf{r}_j) \right\}_{\mathbf{f}=0}^{\mathbf{F}}$$

From now on we use the region *and graph node terms interchangeably as they are linked together and sometimes it simplifies the notation.*

Aiming at having $\mathbf{K}$-(almost)-disjoint paths in $\mathbf{G}$ we define following functions. The function $f : \mathbf{V} \times \mathbf{V} \to <0,1>$ assigns a likelihood that all paths

that are incident on $\mathbf{V}_i$ will continue to $\mathbf{V}_j$. Next we define a decision function $f_T(\mathbf{e}, \mathbf{G})$ which for given edge $\mathbf{e} = (\mathbf{V}_i, \mathbf{V}_j)$, spatio-temporal conditions in graph $\mathbf{G}$ and based on likelihood $f(\mathbf{e})$ either generates a graph $\mathbf{G}'$ by removing some edges (thus $\mathbf{G}' \subset \mathbf{G}$) or leaves it unmodified.

Details on functions f and f_T are described in section 4.2.

Using the f_T function, graphs $\mathbf{G}^0, \mathbf{G}^1, \ldots, \mathbf{G}^n$ are generated such that $\mathbf{G} \supset \mathbf{G}^0 \supset \mathbf{G}^1 \supset \ldots \supset \mathbf{G}^n$. We do this starting from edges with highest likelihood and iteratively continue until no $\mathbf{G}$ modification can be produced due to the low certainty.

Previously described process leads to a sparser graph, mainly in terms of the number of edges. In this $\mathbf{G}^n$ graph, we can observe specific subgraphs which we call *tracklets*. **Tracklet** is a sequence of nodes (regions) - a path in $\mathbf{G}$ with following properties.

Before we describe properties, let us define two sets $\mathbf{E}^-_{\mathbf{V}_i}$, $\mathbf{E}^+_{\mathbf{V}_i}$ and two functions $\delta^-(\mathbf{V}_i)$ and $\delta^+(\mathbf{V}_i)$ and a symbol $\mathbf{P}$. A set of outgoing edges $\mathbf{E}^-_{\mathbf{V}_i} = \{\forall j : (\mathbf{V}_i, \mathbf{V}_j)\}$ and set of ingoing edges $\mathbf{E}^+_{\mathbf{V}_i} = \{\forall j : (\mathbf{V}_j, \mathbf{V}_i)\}$. Vertex out-degree $\delta^-(\mathbf{V}_i) = |\mathbf{E}^-_{\mathbf{V}_i}|$ and in-degree $\delta^+(\mathbf{V}_i) = |\mathbf{E}^+_{\mathbf{V}_i}|$. $\mathbf{P}$ is an ordered set of *regions* - a path in $\mathbf{G}$ and by denoting $\mathbf{P}_i$ we are referencing i-th *region* in this path and $|\mathbf{P}|$ is a path length.

A path $\mathbf{P}$ is a *tracklet* only if it has following fundamental properties:

$$|\mathbf{P}| > 0 \tag{3.1}$$

$$\forall i, i = 1..|\mathbf{P}| : \delta^+(\mathbf{P}_i) = 1 \tag{3.2}$$

$$\forall i, i = 1..|\mathbf{P}| - 1 : \delta^-(\mathbf{P}_i) = 1 \tag{3.3}$$

$$\delta^+(\mathbf{P}_1) \neq 1 \tag{3.4}$$

$$\delta^-(\mathbf{P}_{|\mathbf{P}|}) \neq 1 \tag{3.5}$$

Note that even a single node satisfies the tracklet definition.

The Equation 3.4 and Equation 3.5 implies that a *tracklet* has a maximum number of nodes and cannot be a subgraph/subpath of any other possible *tracklet*.

Let us consider that the segmentation works perfectly, and we get for each frame exactly $\mathbf{K}$ *regions*. If also the rules are faultless, then we get exactly $\mathbf{K}$ *tracklets* - $\mathbf{K}$-disjoint paths - the desired result. Unfortunately, this is not true except for simple cases. Usually, the graph $\mathbf{G}^n$ contains a high number of *tracklets*.

Let us remind that Γ is a set of $\mathbf{K}$-IDs. To overcome an issue with a high number of *tracklets*, a subset $\gamma \subseteq \Gamma$ is assigned to all *tracklets*. This process is called *ID assignment*. γ might be also equal to an empty set. *ID assignment*

starts with obtaining the set of *tracklets* examples $\mathcal{T}$ (at least one *tracklet* per ID). The examples might be provided by the user or extracted automatically. Then an ID classifier is built based on the provided examples. ID-Sets for the rest of *tracklets* are derived based on ID consistency constraints (see section 4.4) combined with classification. The classification is done in a greedy way, starting from the most likely. After each assignment, the constraints are updated and likelihood is reevaluated.

In the end, the *tracklets* with the same ID in their ID-Sets are grouped into a structure called a *track*. When the *tracklets* in each *track* are ordered by frame number, **K**-(almost)-disjoint paths are reconstructed. Paths are not disjoint in frames, where the *region* is shared during object interactions by multiple IDs.

Chapter 4

Approach

4.1 Segmentation Step

In this step, we reduce the search space from the whole image into a set of regions. A perfect segmentation algorithm will return for each frame K segmentations, where **K** is the number of tracked objects. Unfortunately, this is not always the case.

Let us consider simple two thresholds (θ_1, θ_2) segmentation, where we label each pixel (y, x) with intensity $I(y, x)$ as a foreground if $\theta_1 <= I(y, x) <= \theta_2$, otherwise as background. This naive algorithm is a quite common way how the segmentation is treated in others systems[4, 2]. Using this simple method, we will describe and analyze five possible segmentation scenarios which also occur in more sophisticated algorithms.

1. A single object is segmented. We will call this single-id region. This is the ideal case and also the most common one.

2. Oversegmentation is a case when multiple objects are inside one region. We will call this multi-id region.

3. Undersegmentation is a case when one object is divided into multiple regions. We will call this id-part region.

4. A region we are not interested in is segmented (e.g. a pit, border of an arena, food, eggs) and we are not able to eliminate it with segmentation settings. We will call this no-id region.

5. Miss - this is not a result of segmentation as in cases 1-4. It is also relatively rare, but it is possible that the object is not detected (e.g. due to the blur caused by fast movement), and we need to consider this case in the next steps.

An example of segmentation results can be found in Figure 4.3.

We provide two options for segmentation but due to the system modularity it is possible to add more methods in the future.

4.1.1 Maximally Stable Extremal Regions (MSER)

Maximally Stable Extremal Regions (MSER) [15] is a feature detector. The MSER algorithm extracts from an image I a number of co-variant regions, called MSERs. We use this as a default option as it performs well on all video sequences where an object is separable from the background based only on grayscale intensity. Compared to naive one or two thresholds method it provides intensity invariance.

4.1.2 Random Forest Pixel Classifier

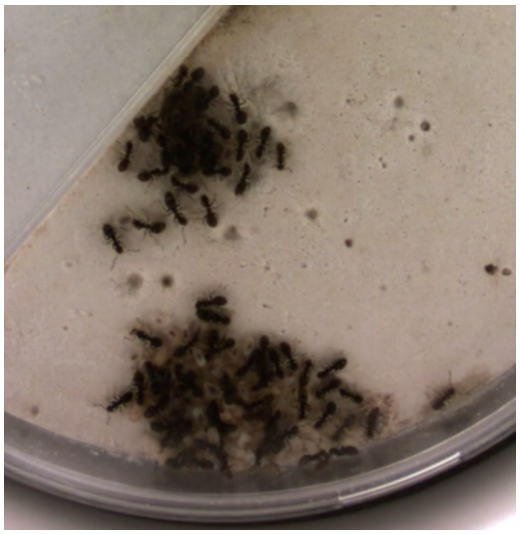


Figure 4.1: An example of a sequence where multiple size objects are present. In this sequence, the segmentation is very challenging. This image has been acquired and kindly made available by Barbara Casillas-Perez, Cremer Group, Institute of Science and Technology Austria.

For more complex scenarios, where objects cannot be segmented using MSER segmentation we provide a tool for training a background/foreground, per pixel classifier. It works in the following way. A random frame is provided, where the user labels sets of pixels as foreground and background. The Random Forest[16] pixel classifier is trained in feature space F, and the provided image is classified. In an iterative way, the user can label more pixels and also choose different frames until they are satisfied with the result.

Using this classifier, a probability of being a foreground is estimated for each pixel. On top of this classification either MSER method or even simple thresholding is applied to obtain segmentations. For more details see section 8.4

4.2 Graph Modifications

In this section, we describe the modifications of the graph formulated in chapter 3. The graph is modified with the goal of minimizing the number of *tracklets* by merging them together. We start by introducing anomaly detection using Isolation Forest in subsection 4.2.1. Then we explain the process of model training data acquisition in subsection 4.2.2 and in sections 4.2.3 and 4.2.4 we describe how to estimate motion and *region* alikeness. Lastly, in subsection 4.2.5 we describe how to combine motion and *region* alikeness to assign the likelihood of each ID set being transferred from one *region* to

another along a given edge - $f(V_i, V_j)$.

Before we start, let us rewaken sets $\mathbf{E}^-_{V_i}$ and $\mathbf{E}^+_{V_i}$. A set of outgoing edges $\mathbf{E}^-_{V_i}$ is defined as $\mathbf{E}^-_{V_i} = \{\forall j : (V_i, V_j)\}$ and set of ingoing edges $\mathbf{E}^+_{V_i} = \{\forall j : (V_j, V_i)\}$.

4.2.1 Isolation Forest

The IsolationForest [17] separates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and the minimum value of the selected feature. Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node. This path length, averaged over a forest of such random trees, is a measure of normality and the decision function. Random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produce shorter path lengths for particular samples, they are highly likely to be anomalies. (*The majority of this paragraph was adapted from Scikit-learn documentation. [18]*)

The anomaly score of an observation sample is computed as $0.5 - 2^{-m/l}$, where m is the average path length of given sample and l is an average path length of all samples from training set. The lower the score, the higher the chance that the sample is an anomaly.

4.2.2 Training Data Acquisition

At first, we remove all edges (u, v) where an ID-Set transfer from *region* u to v is not possible because *regions* are too distant. The object maximum speed and video frame rate are included in one parameter δ - the maximum distance that an object can cover in one frame. For a given δ, all edges (u, v) from graph $\mathbf{G}$ are tested. If the distance between locations u and v is greater than δ, the edge is removed.

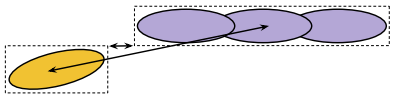


Figure 4.2: Euclidean distance of centroids vs perpendicular distance of *region* circumscribed rectangles.

The *region* distance is measured as a perpendicular distance of *region* circumscribed rectangles. The reason for not using simpler Euclidean distance of centroids is as follows. Imagine a *region* composed of multiple objects in a row and a single object *region* approaching one end of this row. Then the centroid distance might be enormous even if the objects will be close to each other (see Figure 4.2).

After the pruning steps, the graph is usually sparser. Now create a set $E' = \{(u, v)\}$, where $E^-_u = E^+_v = 1$. These pairs are examples of a full ID-Set transfer from u to v and they will serve us as a training dataset for motion and *region* alikeness model.

4.2.3 Motion Model

Now we represent an edge (u, v) from set E' defined in the previous subsection 4.2.2 with three numbers - the centroid distance; change in major axis orientation θ and a result of function $\phi(u, v)$.

$$\theta = \frac{1}{2} \arctan\left(\frac{2\mu'_{11}}{\mu'_{20} - \mu'_{02}}\right),$$

where μ are *region* central moments.

$$\phi(\mathbf{r}_1, \mathbf{r}_2) = \begin{cases} \arccos\left(\dfrac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}\right), & \text{if } \|\mathbf{u}\| > \epsilon \text{ and } \|\mathbf{v}\| > \epsilon \\ 0, & \text{otherwise} \end{cases},$$

where $\mathbf{u} = \mathbf{C}(\mathbf{r}_1) - \mathbf{C}(\mathbf{r}_2)$ (a vector going from centroid of *region* $\mathbf{r}_2$ to the centroid of *region* $\mathbf{r}_1$). $\mathbf{v}$ is a unit vector going from the centroid of $\mathbf{r}_1$ in the direction of *region* major axis. $\epsilon = \frac{\text{average object length}}{10}$ The function ϕ is an heuristic saying that an object is usually moving in the way of major axis, not sidewise. The parameter ϵ triggers the heuristic ϕ only when the object is moving.

In this feature space, an isolation forest is built using training data set E' specified in the previous subsection 4.2.2. Then the edge set E is taken and the anomaly score is computed for each edge. The score is sorted. Highest ten percent of samples are taken as "not an anomaly" examples and ten percent of samples with the lowest score are taken as "anomaly" examples. The score mapping into zero one range is trained using logistic regression [19].

The purpose of this model is to estimate a likelihood of the ID-Set transfer given the movement changes. We mark the likelihood estimator function as $f_m(\mathbf{r}_1, \mathbf{r}_2)$.

4.2.4 Region Alikeness Model

The approach here is the same as in the previous section except for the feature space. The feature space is six-dimensional.

1. $|\mathcal{A}(\mathbf{r}_1) - \mathcal{A}(\mathbf{r}_2)|$, the area difference
2. $\mathcal{A}(\mathbf{r}_1)/\mathcal{A}(\mathbf{r}_2)$, the area ratio
3. $\lambda_1(\mathbf{r}_1) - \lambda_1(\mathbf{r}_2)$, a difference of *region* major axes
4. $\lambda_1(\mathbf{r}_1)/\lambda_1(\mathbf{r}_2)$, a ratio of major axes
5. $(\lambda_1(\mathbf{r}_1)./\lambda_2(\mathbf{r}_1))/(\lambda_1(\mathbf{r}_2)/\lambda_2(\mathbf{r}_2))$, ratio of major and minor axes ratios
6. $|I_{\mathbf{min}}(\mathbf{r}_1) - I_{\mathbf{min}}(\mathbf{r}_2)|$, minimum intensity difference.

The purpose of this model is to estimate a likelihood of the ID-Set transfer given the change in *region* appearance. We mark the likelihood estimator function as $f_a(\mathbf{r}_1, \mathbf{r}_2)$.

4.2.5 Full ID-Set Transfer Decision Function

We design a decision function $f_T(\mathbf{e})$ which for given edge $\mathbf{e} = (\mathbf{r}_1, \mathbf{r}_2)$ decides whether it should be connected into a tracklet or left undecided.

Let us assess a value for all $(\mathbf{r}_1, \mathbf{r}_2) \in E^-_{\mathbf{r}_1}$. The $s(\mathbf{e}) = f_a(\mathbf{e}) \cdot f_m(\mathbf{e})$. We arrange edge and score pairs $p = (\mathbf{e}, s)$ in descending order of score values into a sequence $p_1, p_2, ...p_n$. Notation s_i denotes the score of pair p_i, e_i denotes the edge from pair p_i.

$$f_T(\mathbf{e}) = \frac{s_1}{s_1 + s_2 + \zeta},$$

where ζ is a decision threshold (set to $\zeta = 0.3$). The decision threshold parameter tuning is discused in section 5.1.

When $f_T(\mathbf{e}) > 0.5$, all edges from $E^-_{\mathbf{r}_1}$ are removed from $\mathbf{G}$ except for e_1. The same is applied to all edges from $E^+_{\mathbf{r}_2}$ which merges *tracklets* of $\mathbf{r}_1$ and $\mathbf{r}_2$.

4.3 Region Classification

As we have described in section 4.1, there are five possible results of segmentation. Here we show a way how a *region* could be classified. Once *region* classification is done, the *tracklets* are classified using their most common *region* class. A tracklet class may help with the ID assignment process: no-id *tracklets* could be ignored, id-part could be treated in a particular way, etc. However, mainly the probability distribution of ID presence is estimated in single-id *tracklets*. Based on the probability distribution, ID assignment is performed.

The goal is to sort out the *regions* into four categories. We have tried to design a general classifier but without success. We suggest the following approach.

We do a nearest neighbor (NN) classification [20]. The feature space is seven-dimensional - a *region* area; major axis; minor axis; minimum intensity; maximum intensity; *region* margin; and the length of the *region* contour. Feature space is normalized by subtracting the mean and scaling to unit variance. The metric is Euclidean distance. Next, we ask the user for k-decisions. As the number of k is limited and we want to build the best classifier possible, we will choose k as K-Means++ [21] algorithm does.

The algorithm is as follows:

1. Randomly choose a data point as the first center, using a uniform probability distribution. For each data point x, compute $D(x)$, the distance between x and the nearest center that has already been chosen.

2. Randomly choose a data point as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$.

3. Repeat Steps 2 and 3 until k distinct centers have been chosen.

4. Now that the initial centers have been chosen, display *regions* and let the user assign one of four labels (single-ID, multi-ID, ID-part, no-ID). See examples in Figure 4.3

5. Classify each *region* using 1-NN search.

(Steps 1–3. taken from Wikipedia [22].)

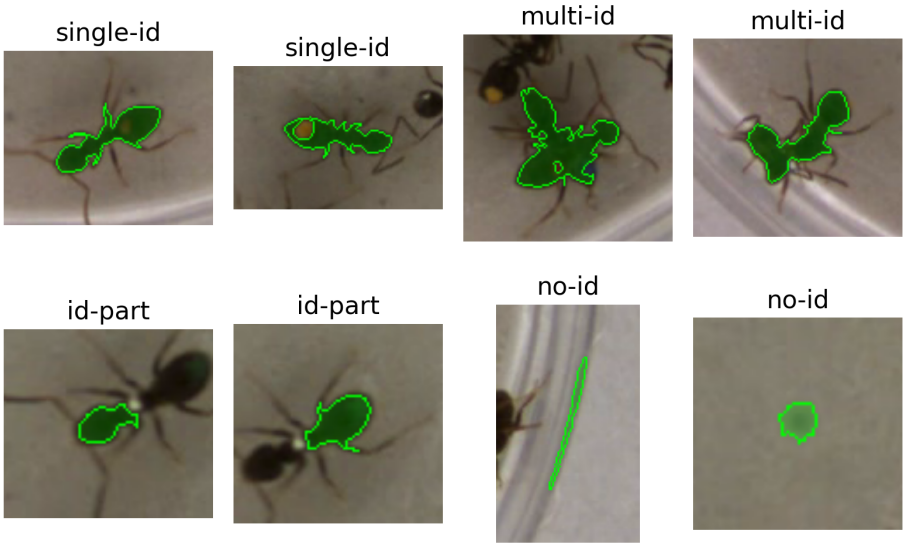


Figure 4.3: Segmentation class examples from Cam1 dataset.

Parameter k tuning and the feature selection is described in section 5.2

4.4 ID assignment Constraints

From the nature of ID-Set emerges that when the identity A is assigned to a tracklet, the A identity cannot be inside any ID-Set of all tracklets with a timespan overlap. In this section, we will define a set of rules called *ID consistency rules (ICR)*. The compliance of *ICR* for all tracklets is a necessary but not sufficient condition for correct *ID assignment*. Let us define a set $\mathbf{T_f}$ as a set of all tracklets passing frame $\mathbf{f}$. The set $\mathbf{F}$ is set of all frames. A tracklet has a set $\mathcal{P}$ and $\mathcal{N}$. By denoting $\mathcal{P}_\mathbf{t}$ we are referring to $\mathcal{P}$ set for tracklet $\mathbf{t}$, likewise for $\mathcal{N}_\mathbf{t}$. The $\mathcal{P}$ set is called ID-Definitely-**P**resent and $\mathcal{N}$ is ID-Definitely-**N**ot-Present. Recalling only that Γ stands for *full ID set*.

Following equations must hold:

$$\forall \mathbf{f} \in \mathbf{F} : \bigcap_{\mathbf{t} \in \mathbf{T_f}} \mathcal{P}_\mathbf{t} = \emptyset \tag{4.1}$$

$$\forall \mathbf{f} \in \mathbf{F} : \bigcup_{\mathbf{t} \in \mathbf{T_f}} \mathcal{P}_\mathbf{t} = \Gamma \tag{4.2}$$

$$\forall \mathbf{t} : \mathcal{P}_{\mathbf{t}} \cap \mathcal{N}_{\mathbf{t}} = \emptyset \tag{4.3}$$

We say that ID-Set for tracklet $\mathbf{t}$ is solved when

$$\mathcal{P}_{\mathbf{t}} \cup \mathcal{N}_{\mathbf{t}} = \Gamma \tag{4.4}$$

4.5 ID assignment Detection

Given training set $\mathcal{T}_0 = \{(\mathbf{t}_i, ID)\}$, the Random Forest Classifier (RFC) is trained (used feature space and classifier parameter tuning is described in section 5.1). We compute ID probability distribution for each tracklet classified as Single-ID. Then we iterate until each tracklet is labeled:

1. order *tracklets* by decision probability and pick best pair d = $(\mathbf{t}_j, ID)$
2. if the certainty < certainty threshold ask expert and follow step E1), else continue to step 3
3. assign ID to $(\mathbf{t}_j)$
4. check ID set consistency rules. If inconsistency is found raise question and follow step E2)
5. update classifier with a new example, update decisions probability and go to step 1)

E1) A question is raised, training set $\mathcal{T}_{i+1} = \mathcal{T}_i \cup answer$. Go to step 5.
E2) Raise a question as in E1 and update training set. Then remove all training examples not created by the expert and also roll back all automatic decisions. Then go to step 5.

Step E2 guarantees the convergence of this algorithm because after each iteration the training set $\mathcal{T}$ is increased until all *tracklets* are labeled without causing ID sets inconsistency. It means that in the worst case, the expert has to label each tracklet by hand.

The ID probability distribution is estimated using a random forest classifier[16]. The random forest classifier is a forest of decision trees. A decision tree is a tree where a feature and a threshold are defined in each node. The sample classification is based on training samples distribution in leaves where all decisions have been made. The trees are trained on a random sample of the training set, and split decisions are done on a random subset of a feature set. The forest classification is computed as an average of all trees classifications. The random training set and random feature subset contribute against overfitting.

The ID probability distribution is estimated for all *regions* of a given tracklet. The IDs denied by ID-Consistency rules are omitted in this process. The tracklet ID distribution is then a mean of all *region* distributions. Let us

denote by $\mathbf{P}^{\gamma}(\mathbf{t})$ the probability of γ on a given tracklet $\mathbf{t}$. $\mathbf{P}_1(\mathbf{t})$ stands for the greatest probability from probability distribution over identities for a given tracklet and $\mathbf{P}_2$ is the second greatest. The certainty $\mathbf{c}$ of an γ-Assignment is modeled as

$$\mathbf{c} = (1 - \alpha) \cdot 0.5 + \alpha \cdot \frac{\mathbf{P}^{\gamma}(\mathbf{t})}{\mathbf{P}_1 + \mathbf{P}_2}.$$

$$\alpha = \min\left((\frac{|\mathbf{t}|}{\kappa})^2, 0.99\right),$$

where $|\mathbf{t}|$ is the tracklet length and κ is a tuning parameter, for our experiments we set κ to 50. User can configure a threshold $\mathbf{C}$, and once the best possible assignment certainty $\hat{\mathbf{c}} < \mathbf{C}$ the user is prompted to provide an annotation. When $\mathbf{C} = 0$, the system will not raise any questions and returns the best possible results. When $\mathbf{C} = 1$, the system asks about all *tracklets* except for those automatically resolved by ICR.

4.6 Expert Annotation

An expert annotation is an input provided by the user as a reaction to a raised question (e. g. is the identity in the tracklet one same as in the tracklet two; which identity is in this tracklet). This led us to the domain of supervised learning, visualization and Human-Computer Interface (HCI). Several questions emerge. What is a good annotation? How to lay and choose questions and what is an impact of obtained information? What is a complexity and time consumption of an annotation? How to support the user when dealing with decisions during annotation process?

In our opinion, a good annotation is a user provided information which is correct and in the given situation has the highest information gain (e. g. an ID-Decision for the longest tracklet with ambiguous ID-Set was provided; a pair of timespan overlapping *tracklets*, with similar ID distribution is differentiated; the correction when inconsistency of ID-Set-Constraints occurs). From the user's point of view, a good annotation is such an annotation which is not time-consuming. The annotation process differs based on whether the user is capable of ID re-detection when a random frame is shown. If the re-detection is not possible, then the space of possible questions shrinks to questions about ID-Set equality of tracklet pairs in almost consecutive frames. For example, two objects, let us call them A nad B, gathers and the interaction starts and the identity is lost. Once the interaction ends and they are separated again, the expert matches current *tracklets* to A and B before interaction.

Chapter 5

Parameters Tuning

5.1 Decision Threshold - ζ

The parameter ζ is used in Full ID-Set Transfer Decision Function, described in subsection 4.2.5. It's domain range is <0,1>. The higher the parameter is the more conservative the system is in decisions about the CToIS (Complete Transfer of ID-Set) thus the tracklet number will raise while the length of tracklets will drop. If the number is too low, then a false tracklet might be created, which produces errors during ID-Assignment as the ID-Consistency Rules are violated. Keeping this in mind the ζ was set by hand to 0.3.

5.2 Number of labeled samples for *Region* Classification training

We have measured an impact of the number of provided labeled samples - k, on *region* classification described in section 4.3. Firstly we have created a ground truth from thousand most distant *regions* for each dataset. The reason for poorer performance on Cam1 and Camera3 is that segmentation sometimes includes ant's antennas and legs. This decreases data separability (e. g. Single-id *region* segmented with legs will not differ much in area and contour length to Multi-id *region* segmented without legs).

Based on our measurements we suggest using k = 50 and full feature set (a *region* area; major axis; minor axis; min intensity; max intensity; *region* margin; and the length of the *region* contour) as it shows the highest stability while having good performance. (see Figure 5.1 and Figure 5.2, the full set is labeled as f4). In our experience, labelling 50 samples takes less than one minute.

We are aware of the incompleteness of this approach. It is a temporary solution which should be replaced with more robust and appropriate semi-supervised learning techniques.

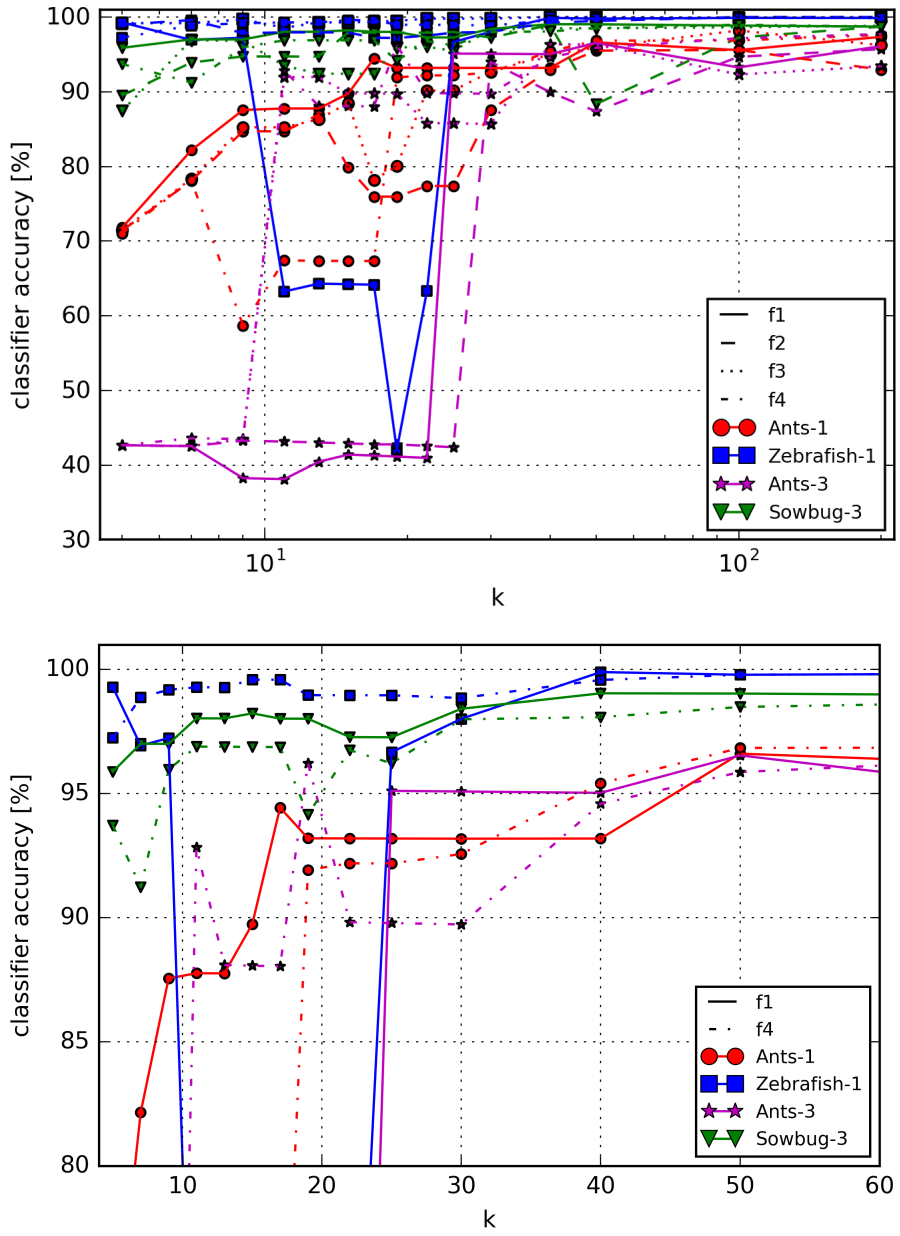


Figure 5.1: An impact of a number of **k**-diversely labeled samples and a feature sets on 1-NN *region* classifier accuracy. Feature set f1 = {region area, major/minor axes length, contour length}, f2={f1, *region* min. intensity}, f3 = {f2, *region* max. intensity}, f4 = {f3, *region* margin}

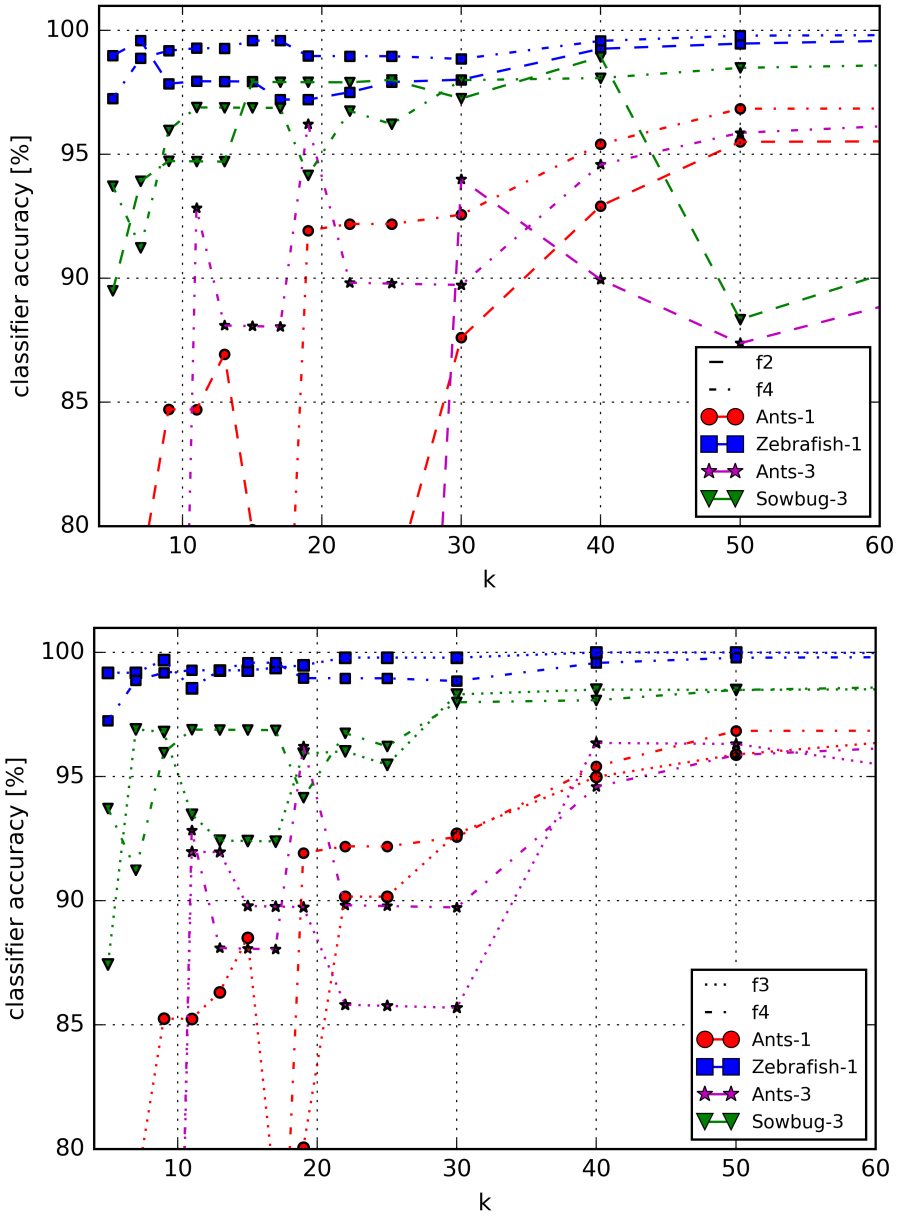


Figure 5.2: An impact of a number of **k**-diversely labeled samples and a feature sets on 1-NN *region* classifier accuracy. Feature set f1 = {region area, major/minor axes length, contour length}, f2={f1, *region* min. intensity}, f3 = {f2, *region* max. intensity}, f4 = {f3, *region* margin}

5.3 Random Forest Classifier (RFC) - Parameter Tuning

In this section, we describe different Random Forest Classifier parameters and their effect on classifier accuracy. We tune one parameter at the time comparing to default values. The performance is evaluated on various features (described in section see subsection 5.3.1) and different datasets (see chapter 6). The results are shown in a table of performance differences.

The evaluation has been done ten times for each configuration, dataset, and feature descriptor. The mean accuracy was computed and is displayed in succeeding tables. The classifier was trained on 5% randomly picked single-id *regions* and evaluated on the rest. Thus there were 10 training sets and 10 test sets.

We have tried the evaluation for different sizes of a training dataset (1%, 2%, 5%, 10%, 15% and 20%) but the relations between results were more or less the same, shifting up and down by a constant. Thus we are showing only results for 5% as it is close to the size of a training set in practice.

The parameters we were tuning are - a node split **criterion**; **max-f** - maximum number of features used to find a split; **min-leaf** - minimum of samples in a leaf; **n-trees** - number of estimators; **max-d** - the maximum depth of a decision tree.

The default values were a node split **criterion** = entropy; **max-f**= 50%; **min-leaf** = 1; **n-trees**= 10; **max-d** = unlimited.

	Ants-1	**Zebrafish-1**	**Ants-3**	**Sowbug-3**
Moments	74.52%	64.64%	58.21%	81.59%
Colornames	74.13%	55.09%	60.51%	75.17%
C-co-occurence	84.50%	79.05%	75.78%	70.47%
I-co-occurence	87.27%	82.42%	73.61%	74.46%
LBP	43.51%	35.29%	31.96%	62.22%
HoG	43.76%	35.25%	31.97%	61.45%

Table 5.1: Random Forest Classifier performance default values set to: **criterion** = entropy; **max-f** = 50%; **min-leaf** = 1; **n-trees** = 10; **max-d** = unlimited. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.. All following results are compared with this table. Better performance is highlighted in green.

5.3.1 Features

We have tried six different feature descriptors for ID detection. In this section, we briefly introduce them and show their performance.

Moments

We have designed a simple, fast *region* descriptor with dimensionality 19. It is composed of:

- (2) *region* area, contour length
- (3) major axis, minor axis, major/minor axis ratio
- (7) Hu moments[23] of a *region* (binary image)
- (7) Hu moments[23] of a grayscale image masked by *region*

(Major and minor axis are axes of an ellipse interpolating the region.*)*

We suggest usage of this descriptor when the computational cost of others is a problem or if the object shape and size differs dramatically.

Colornames

Weijer et al. introduced a pixel classifier [24] trained on Google image dataset. Pixels are classified into eleven color classes using a RGB look-up table. Using this classifier, all pixels from the *region* are labeled. Then a frequency of all classes is computed. The frequencies are computed on different scale levels. First on the whole *region*, then on a 2x2 grid and then on 4x4 grid. All these eleven bins histograms are vectorized into a description vector of length 231.

We suggest usage of this descriptor for sequences where a lot of color information is present.

Intensity-co-occurrence matrix (I-co-occurrence)

Perez et al. showed a usage of contrast and intensity co-occurrence matrix for animal identity description [4]. An intensity level co-occurrence matrix is a histogram of a co-occurring sum of grayscale values at a given offset over an image. For an 8bit image, it is 512 times max offset matrix. They have classified their objects using nearest neighbor search. The distance metric was simple mean of matrix per element difference. We are using the same matrices for description. The max offset is set to an average object length (2 times major axis of an ellipse interpolating an object). The classification is done using random forest classifier on vectorized co-occurrence matrices.

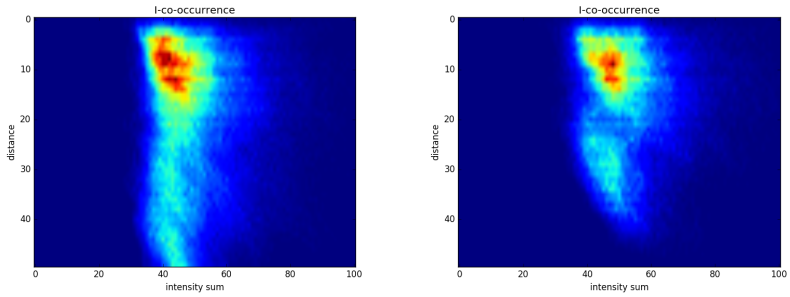


Figure 5.3: An examples of I-co-occurrence matrices for 2 different object identities

Contrast-co-occurrence matrix (C-co-occurrence)

The same as Intensity-co-occurrence matrix in previous section 5.3.1 except for the shape. The co-occurrence matrix here is an absolute value of grayscale values difference at a given offset over an image. Thus the matrix shape is 256 times max offset.

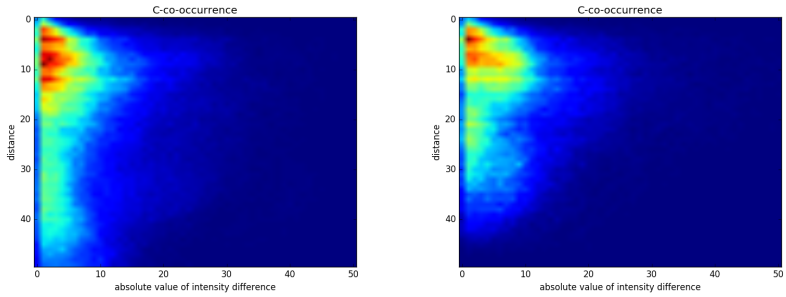


Figure 5.4: An examples of I-co-occurrence matrices for 2 different object identities

Local Binary Pattern (LBP)

The LBP [25] descriptor is commonly used in a pattern recognition problems introduced in 2007 by Zhang et al. We have designed our descriptor in following way. An image is rotated with a rotation center in the *region* centroid with a purpose of aligning all *regions* main axes. Then an image is cropped around a *region*. The LBP descriptor is computed in this crop. Then the 32 bins histogram is counted on the whole image and in 3x1 grid. This is done twice with different LBP parametrization. First with P=24 and R=3 and the second P=8, R=1. Where P is a quantization of the angular space. R is the radius of a circle (spatial resolution). These histograms are concatenated into a vector of length 256.

HoG

The HoG is an another commonly used pattern recognition method introduced in 2005 by Dalal et al. [26]. The image is prepared the same way as for LBP descriptor described in preceding subsubsection 5.3.1. There are several histograms computed. First one on a whole image and then on a grid of size 4x1. The number of orientations was set to eight. Thus the result vector's size is 40.

5.3.2 Split Criterion Function

We were deciding between two split quality functions. The Gini impurity and the information gain - the entropy. Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. The

information gain (also known as Mutual Information) is a ratio of information gain to the inrinsic information.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-0.24%	-0.03%	**+0.17%**	-0.41%
Colornames	**+1.08%**	-0.25%	**+0.11%**	-0.92%
C-co-occurence	-0.20%	-1.08%	-1.05%	-0.70%
I-co-occurence	-0.03%	-2.52%	-1.05%	-0.30%
LBP	**+0.52%**	-0.11%	**+0.16%**	-0.55%
HoG	**+0.57%**	**+0.12%**	**+0.31%**	**+0.24%**
An average	**+0.28%**	-0.65%	-0.23%	-0.44%

Table 5.2: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **criterion** = gini. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

Based on the comparison shown in Table 5.2, we have decided to use the information gain as a split criterion function.

5.3.3 Max Depth of a Decision Tree

The **max-d** influences overfitting as it restricts the maximum depth of all decisions trees in a forest. We have evaluated the parameter in the range <5, 50>. One must be careful with setting it too low as it then tends to underfit. Setting **max-d** ≥ 20 produced zero change to default setting. Here we show only the result of **max-d** set to 10 as this is the value that was picked due to the best results. Full results can be found in Appendix A.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-0.20%	**+0.52%**	**+0.09%**	**+0.18%**
Colornames	-0.17%	**+0.08%**	**+0.45%**	-0.03%
C-co-occurence	0.00%	**+0.01%**	0.00%	**+0.12%**
I-co-occurence	0.00%	-0.03%	0.00%	**+0.12%**
LBP	**+0.26%**	**+0.29%**	-0.23%	**+0.06%**
HoG	**+0.18%**	**+0.41%**	**+0.08%**	-0.17%
An average	**+0.01%**	**+0.21%**	**+0.07%**	**+0.05%**

Table 5.3: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-d** = 10. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

5.3.4 Min Samples In a Leaf

The **min-leaf** defines minimum samples in a leaf. We have evaluated this parameter in the range <1, 5>. Usually, the default value is set to one but

we have found that a slightly better results can be obtained when using a higher value. We have decided to set **min-leaf** = 3. All results are placed in Appendix A.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-0.48%	**+0.84%**	-0.30%	-0.75%
Colornames	-0.87%	**+0.41%**	**+0.46%**	-0.32%
C-co-occurence	-0.26%	**+0.58%**	**+0.05%**	**+0.18%**
I-co-occurence	-0.08%	-0.13%	-0.14%	**+0.06%**
LBP	**+1.06%**	**+0.98%**	**+0.77%**	**+0.86%**
HoG	**+1.29%**	**+0.77%**	**+0.57%**	**+0.23%**
An average	**+0.11%**	**+0.58%**	**+0.24%**	**+0.04%**

Table 5.4: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **min-leaf** = 3. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

5.3.5 Max Features

The **max-f** defines the number of randomly selected features to compute the split. We have evaluated the parameter in the range of <0, 100>% and for the $\sqrt{f}$, where f is the number of features. If the number is too low and the number of weak features is high, the RFC tends to have poorer performance. If it is too high, the RFC tends to overfit. Usually, the default value is set to $\sqrt{f}$. Based on measured performance, we have decided to set a **max-f** = 50% as the average performance of others was lower. We are not displaying the table, because 50% was the default value, thus the difference is 0. We are showing 40% and 60%. All results are presented in Appendix A.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-0.36%	**+0.15%**	**+0.13%**	-0.07%
Colornames	-0.13%	-0.10%	**+0.34%**	**+0.10%**
C-co-occurence	**+0.04%**	-0.01%	**+0.07%**	**+0.26%**
I-co-occurence	-0.16%	-0.04%	-0.10%	**+0.27%**
LBP	**+0.43%**	-0.15%	-0.78%	-0.43%
HoG	**+0.02%**	**+0.05%**	-0.21%	-0.33%
An average	-0.03%	-0.02%	-0.09%	-0.03%

Table 5.5: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 40. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	**+0.07%**	**+0.11%**	**+0.41%**	-0.40%
Colornames	-0.26%	-0.29%	**+0.14%**	-0.04%
C-co-occurence	**+0.06%**	-0.00%	-0.09%	-0.05%
I-co-occurence	-0.14%	-0.18%	-0.24%	-0.07%
LBP	**+0.23%**	**+0.28%**	-0.05%	-0.09%
HoG	**+0.41%**	**+0.30%**	**+0.39%**	**+0.01%**
An average	**+0.06%**	**+0.04%**	**+0.09%**	-0.11%

Table 5.6: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 60. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

5.3.6 Number of Decision Trees

Here was confirmed what usually holds for RFC that higher the number of estimators the better results are obtained. We are using 100 decision trees due to the computational cost during RFC training, but when a performance is not an issue, we suggest going to 200 even higher. All results are placed in Appendix A.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	**+2.63%**	**+2.85%**	**+3.47%**	**+2.37%**
Colornames	**+3.81%**	**+2.95%**	**+8.44%**	**+3.47%**
C-co-occurence	**+1.91%**	**+4.86%**	**+5.68%**	**+4.76%**
I-co-occurence	**+1.69%**	**+4.96%**	**+6.34%**	**+4.37%**
LBP	**+10.91%**	**+6.71%**	**+12.67%**	**+11.04%**
HoG	**+10.54%**	**+6.55%**	**+10.05%**	**+7.39%**
An average	**+5.25%**	**+4.81%**	**+7.77%**	**+5.57%**

Table 5.7: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 100. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	**+2.80%**	**+2.98%**	**+3.57%**	**+2.44%**
Colornames	**+4.17%**	**+3.11%**	**+9.05%**	**+3.60%**
C-co-occurence	**+1.94%**	**+5.10%**	**+6.04%**	**+5.35%**
I-co-occurence	**+1.77%**	**+5.25%**	**+6.73%**	**+4.64%**
LBP	**+11.88%**	**+7.16%**	**+14.36%**	**+12.06%**
HoG	**+11.84%**	**+7.26%**	**+11.09%**	**+7.82%**
An average	**+5.73%**	**+5.14%**	**+8.47%**	**+5.98%**

Table 5.8: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 200. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

5.4 RFC Overall Setting

Based on the previous experiments we have decided to use RFC with the following parameters: a node split **criterion** = entropy; **max-f** = 50%; **min-leaf** = 3; **n-trees** 100; **max-d** = 10. Next we show the overall boost of a classifier for single features (in Table 5.9) and a performance of single feature set compared to mixture of all in Table 5.10.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	**+3.50%**	**+2.48%**	**+3.67%**	**+2.64%**
Colornames	**+3.64%**	**+2.27%**	**+7.84%**	**+4.30%**
C-co-occurence	**+4.51%**	**+4.39%**	**+4.81%**	**+4.13%**
I-co-occurence	**+3.78%**	**+4.17%**	**+5.54%**	**+4.73%**
LBP	**+12.20%**	**+5.47%**	**+9.67%**	**+8.71%**
HoG	**+11.91%**	**+5.37%**	**+8.60%**	**+6.98%**
An average	**+6.59%**	**+4.02%**	**+6.69%**	**+5.25%**

Table 5.9: Tuned RFC performance difference compared to default settings.

RFC tuned	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	78.02%	67.11%	61.89%	84.22%
Colornames	77.77%	57.35%	68.36%	79.47%
C-co-occurence	89.01%	83.44%	80.59%	74.60%
I-co-occurence	**91.05%**	86.58%	79.15%	79.19%
LBP	55.71%	40.76%	41.63%	70.94%
HoG	55.67%	40.61%	40.58%	68.43%
All	89.53%	**88.76%**	**84.20%**	**84.98%**

Table 5.10: The overall comparison of tuned RFC on different feature sets. The last row shows tuned RFC performance, when the combination of all features was used.

Chapter 6

Dataset

In the following text, we introduce four video sequences. We have created a ground truth for all presented datasets. Ground truth is a set of (x, y) coordinates for an identity in every frame. The creation process was based on results of *FERDA* system - so the (x, y) coordinates are centroids of MSERs. When identities were swapped, we have fixed it. During interaction, we have combined the hand corrections with interpolation between provided positions. At the end of the creation process, the ground truth was visually checked.

name	**#frames**	**GT**	**#objects**	**#px**	FPS
Sowbug-3	4500	1-4500	5	407	29.93
Ants-1	4500	1-4500	6	775	15
Ants-3	156 504	1-4500	10	817	15
Zebrafish-1	15 000	1-5000	5	781	32.22

Table 6.1: Dataset overview. In column GT is the frame interval covered by ground truth. In column #px is the average object pixel resolution.

6.1 Camera3

There are ten ants in a petri dish. This video has been acquired and kindly made available by Barbara Casillas-Perez, Cremer Group, Institute of Science and Technology Austria. This dataset has best pixel resolution per animal amongst all our datasets (see Table 6.1. Difficulties may occur during segmentation where an abdomen might be disconnected from the body due to the thickness of waist.

Frame samples from this dataset are shown in Figure 6.1

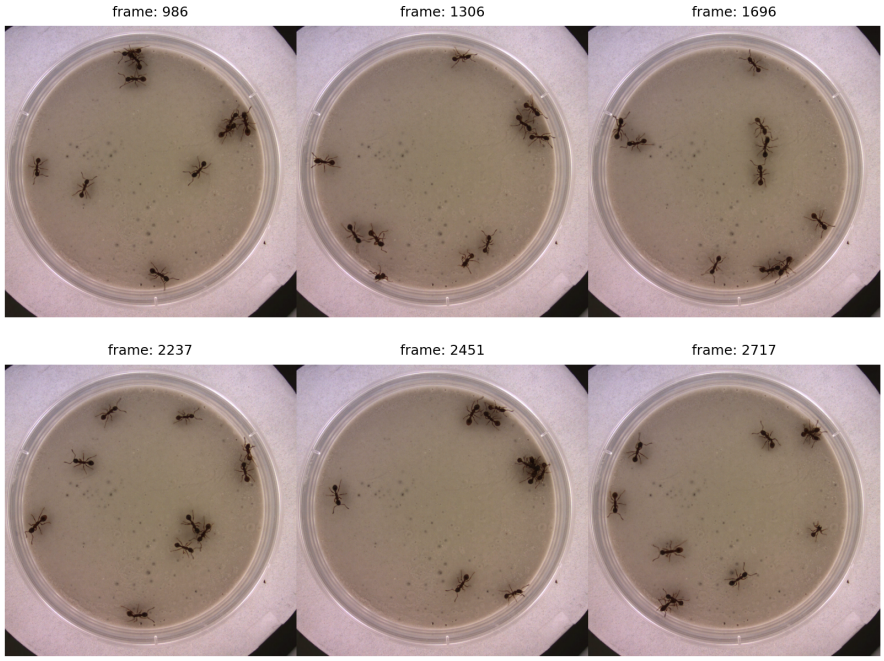


Figure 6.1: Camera3 dataset frame samples.

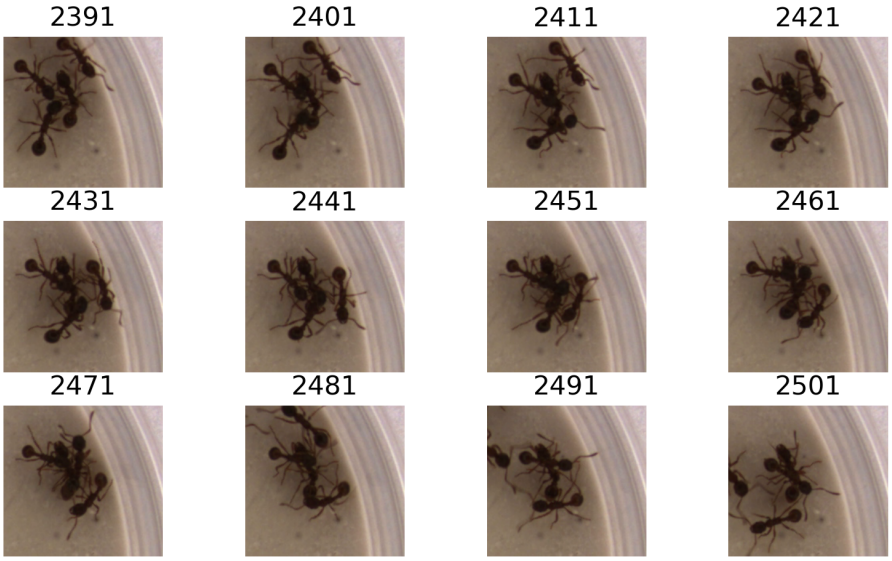


Figure 6.2: An interaction example sequence in dataset Ants-3.

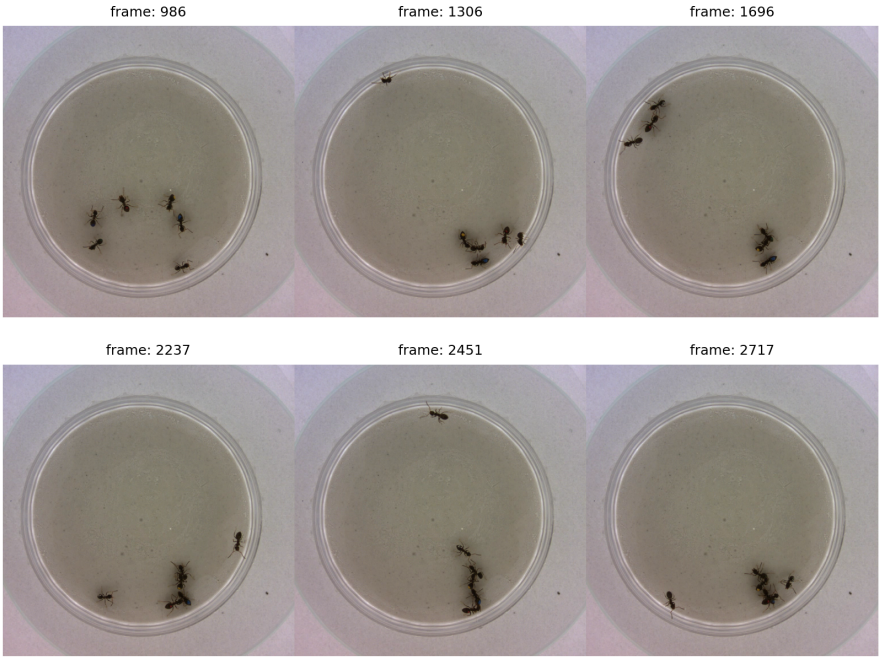


Figure 6.3: Cam1 dataset frame samples.

6.2 Cam1

This video has been acquired and kindly made available by Barbara Casillas-Perez, Cremer Group, Institute of Science and Technology Austria. This dataset is peculiar in that the ants have a color dots on their abdomens. This color aid is quite frequent in biology applications mainly when biologists need to measure some properties (e.g. amount of fungal spores, immune gene expression) after an experiment.

Frame samples from this dataset are shown in Figure 6.3

6.3 Zebrafish

This is a public dataset available on *idTracker* [27] websites. There are five zebrafish in a shallow aquarium.

One of the benefits of this video is that it tests the system's adaptability to morphological changes which are bigger compared to ants or bugs.

Frame samples from this dataset are shown in Figure 6.4

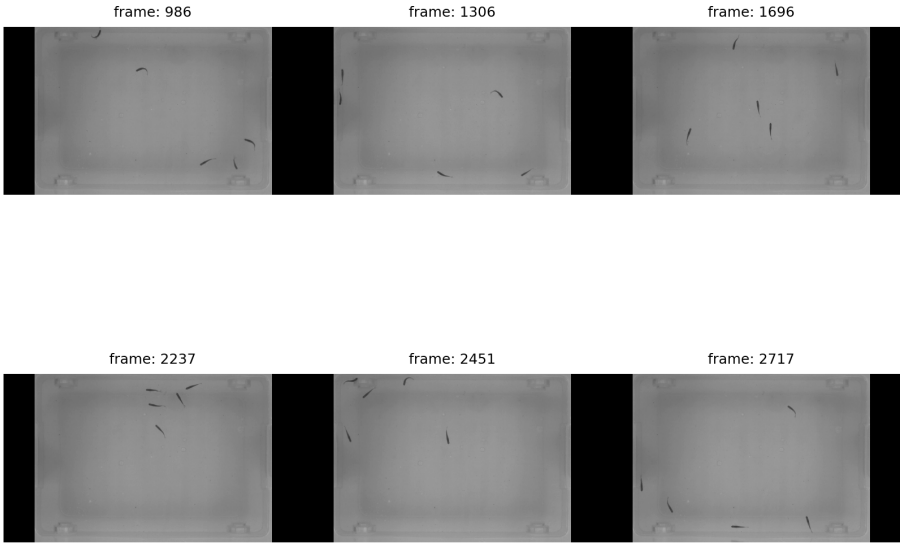


Figure 6.4: Zebrafish dataset frame samples.

6.4 Sowbug3

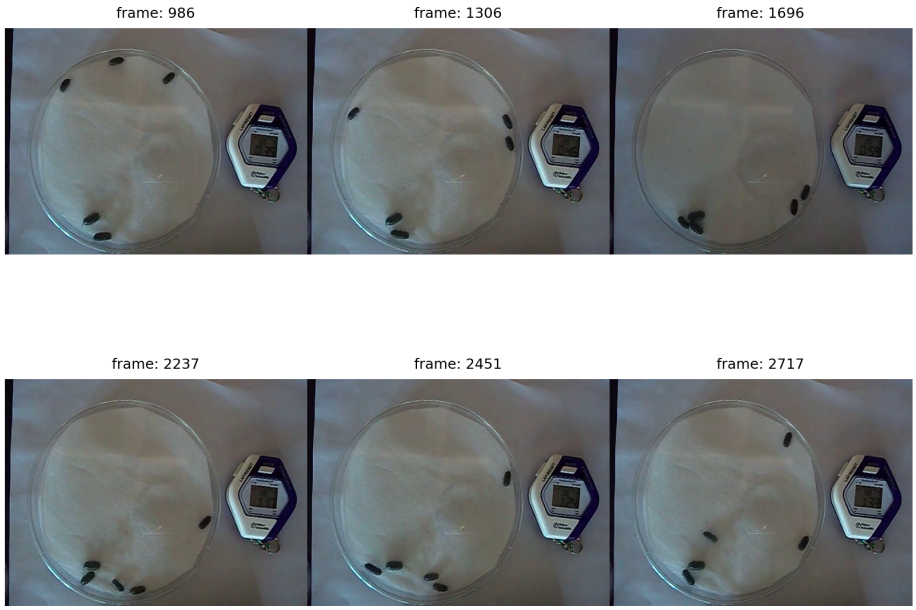


Figure 6.5: Sowbug3 dataset frame samples.

There are five woodlouse (sowbugs) in a petri dish. This sequence is interesting for three reasons. Firstly, it was recorded in an uncontrolled environment. Thus the light conditions are not homogeneous. Secondly,

FERDA was originally developed mainly for ant tracking purposes and thus, we put to test the performance on an animal with distinct morphology and motion behaviour than ants. Lastly, the object size is smaller than in other datasets. This yields challenges for both the segmentation and ID assignment steps and it serves as a validation process.

Due to the hand oclussion, petri dish movements and object absence in the beginning and the end of a video, we are using only video crop. (Crop starts around 27s and lasts for 150 seconds). The dataset was obtained from YouTube [28]. Unfortunately, we were not successful in contacting the author thus we are not publishing this sequence.

To allow others to replicate the same settings, we provide exact parameters we used to cut the video:

```
ffmpeg -ss 26.9295021717 -t 150.3508185767 -i Sowbug3.mp4 -an
-vcodec rawvideo -async 1 Sowbug3-crop.avi
```

Frame samples from this dataset can be seen in Figure 6.5

Chapter 7

Experiments

In this chapter, we show *FERDA* performance on different species: ants, fish and bugs in comparison with *idTracker*, the current state of the art animal tracking software.

7.1 ID-Classifier Comparison

In this experiment, the RFC tuned in section 5.3 on single feature sets was tested on all features and compared with default RFC parameters and Nearest Neighbor classifier used in *idTracker*. The NN was searched twice: for I-co-occurrence matrix and for C-co-occurrence matrix. The metric for NN was the mean of absolute values of per-pixel differences. If the NN ID for Intensity and Contrast co-occurrence matrix disagreed, the decision was not made. (This is how it is described in *idTracker* paper's Supplementary Note 2.) The training and tests sets were obtained the same way as described in section 5.3. The evaluation of each classifier was done ten times on each dataset, and the accuracy was averaged. The results of this experiment are shown in Table 7.1.

	Ants-1	**Zebrafish-1**	**Ants-3**	**Sowbug-3**
NN	84.82%	70.18%	70.71%	77.47%
RFC default	88.03%	85.63%	78.75%	80.58%
RFC tuned	**89.53%**	**88.76%**	**84.20%**	**84.98%**

Table 7.1: Overall comparison of different classifier's performance in ID-Assignment task.

7.2 FERDA's Overall Performance

In this experiment, the object trajectories provided by our system and *idTracker* were compared with a Ground Truth (GT). The (x, y, ID) triplets in each frame were matched to the nearest GT (x', y', ID') triplet within a radius of one-third of an average object body length. A permutation of identities provided by GT onto those provided by the systems was sought such that the number of correctly assigned IDs is maximized. The system

accuracy is computed as the sum of correctly assigned IDs divided by the number of all GT entries. The error is computed analogically. GT triplets with no object body in the radius are considered unassigned.

IdTracker provides two possible kinds of output, either detected (x, y, ID) triplets with raw position or, alternately, an augmented position interpolated between two nearest tracklets (in *idTracker* terminology fragments) with the same ID. We found that these interpolated results are imprecise mainly when the tracklets frame distance is high. In our comparison, we are showing both the basic and the interpolated results (in tables denoted as I).

7.2.1 With No Expert Annotations

These experiments were run with Expert Annotation turned off. The training set for ID-Assignment is found using the following algorithm. All sets of tracklets having at least one frame in common are found. If any of the tracklets in such a set is classified other than Single-ID, then the set is omitted; otherwise the set is characterized by the length of its shortest tracklet. The set with maximum shortest tracklet length is picked as the training set $\mathcal{T}$. This training set is not augmented with new examples as described in section 4.5, because when an inconsistency is detected, there is no mechanism implemented to learn from own mistakes (but we think that learning from mistakes is possible and it will be one of the topics for future work).

Without ID-Consistency Rules

		FERDA	idTracker	idTracker I
Ants-1	*correct*	59.66%	71.68%	**95.95%**
	wrong	6.74%	**0.32%**	1.61%
	unassigned	33.60%	28.01%	2.44%
Zebrafish-1	*correct*	**96.92%**	88.00%	94.02%
	wrong	**0.27%**	0.63%	3.58%
	unassigned	2.81%	11.36%	2.40%
Ants-3	*correct*	61.24%	82.38%	**90.43%**
	wrong	25.60%	**5.25%**	5.80%
	unassigned	13.17%	12.37%	3.77%
Sowbug-3	*correct*	56.22%	70.60%	**76.06%**
	wrong	30.72%	**15.28%**	17.31%
	unassigned	13.06%	14.12%	6.62%

Table 7.2: Results of tracking system performance on four datasets described in chapter 6. The *idTracker* column is a regular idTracker output, *idTracker I* is an output with provided interpolation.

With ID-Consistency Rules

		FERDA	idTracker	idTracker I
Ants-1	*correct*	59.50%	71.68%	**95.95%**
	wrong	6.81%	**0.32%**	1.61%
	unassigned	33.69%	28.01%	2.44%
Zebrafish-1	*correct*	**96.92%**	88.00%	94.02%
	wrong	**0.27%**	0.63%	3.58%
	unassigned	2.81%	11.36%	2.40%
Ants-3	*correct*	68.80%	82.38%	**90.43%**
	wrong	19.04%	**5.25%**	5.80%
	unassigned	12.17%	12.37%	3.77%
Sowbug-3	*correct*	70.30%	70.60%	**76.06%**
	wrong	**14.20%**	15.28%	17.31%
	unassigned	15.50%	14.12%	6.62%

Table 7.3: Results of tracking system performance on four datasets described in chapter 6. The *idTracker* column is a regular idTracker output, *idTracker I* is an output with provided interpolation.

7.2.2 With Expert Annotations

Here, we show what happens when an Expert is asked to add more data to the training set. The training set $\mathcal{T}$ is obtained the same way as it was described in the previous section. This usually means that the number of region examples per ID differs as the provided tracklets vary in size. We prompt the user for input until each tracklet consists of more than N regions. Training set refilling is done in an iterative way.

1. Choose the ID with the lowest number of examples n
2. If n $>$ N exit, else go to 3
3. Order all tracklets in $\mathcal{T}$ that contain the chosen ID by time. Choose the latest one.
4. Play a video starting ten frames before the end of the chosen tracklet (usually, the end is caused by an interaction).
5. Let the user control the player (e. g. framerate) to let them investigate and find a tracklet that contains the same object, extending the original tracklet.
6. Update set $\mathcal{T}$ and go to 1.

The whole user interaction usually takes less than 5 seconds per tracklet. In this experiment, we set N to one thousand regions per tracklet. As segmentation difficulty and the layout of interactions varies in different sequences, the number of needed annotations differs significantly. It was 1 for Sowbug-3

sequence, 48 for Ants-1, 30 for Ants-3 and 6 for Zebrafish. The results are shown in Table 7.4 and in Figure 7.1, 7.2.

Figure 7.1: ID detection results on Sowbug-3, Ants-1, Ants-3, Zebrafish-1 datasets. Y axis labels identities, X axis labels frames. Top row - idTracker (**without** interpolation), middle (thinner) row - ground truth, bottom row - FERDA (**with** expert help during initialisation). Gray - ID unassigned.

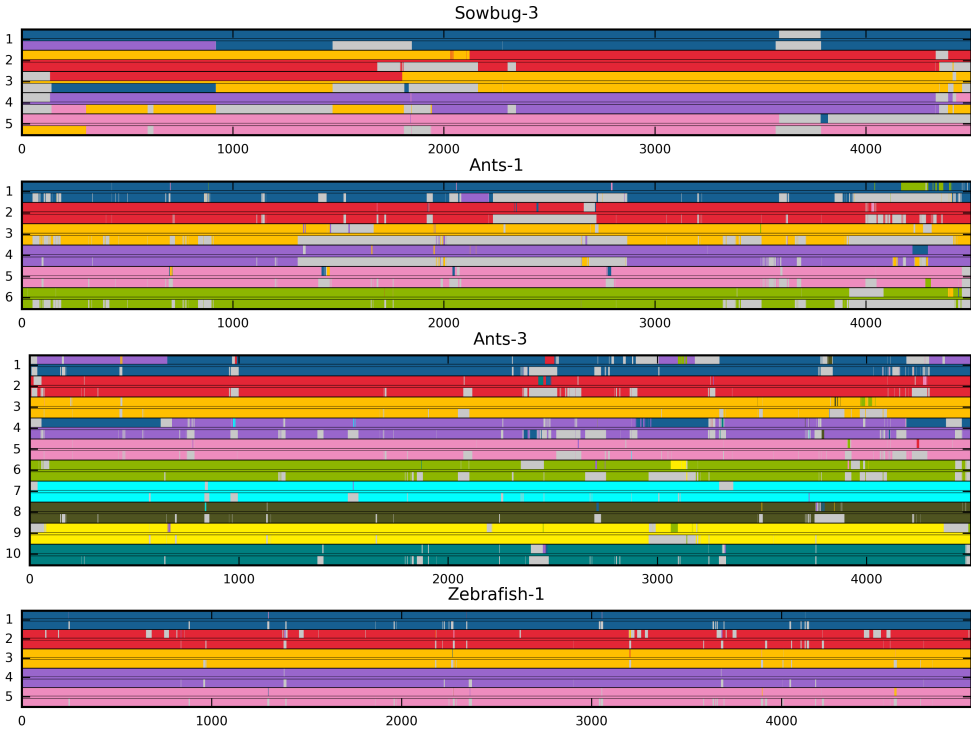


Figure 7.2: ID detection results on Sowbug-3, Ants-1, Ants-3, Zebrafish-1 datasets. Y axis labels identities, X axis labels frames. Top row - *idTracker* (**with** interpolation), middle (thinner) row - ground truth, bottom row - FERDA (with expert help during initialisation). Gray - ID unassigned.

		FERDA	**idTracker**	**idTracker I**
Ants-1	*correct*	67.23%	71.68%	**95.95%**
	wrong	1.57%	**0.32%**	1.61%
	unassigned	31.20%	28.01%	2.44%
Zebrafish-1	*correct*	**97.27%**	88.00%	94.02%
	wrong	**0.00%**	0.63%	3.58%
	unassigned	2.73%	11.36%	2.40%
Ants-3	*correct*	87.80%	82.38%	**90.43%**
	wrong	**0.78%**	5.25%	5.80%
	unassigned	11.42%	12.37%	3.77%
Sowbug-3	*correct*	71.58%	70.60%	**76.06%**
	wrong	**13.67%**	15.28%	17.31%
	unassigned	14.74%	14.12%	6.62%

Table 7.4: Results of tracking system performance on four datasets described in chapter 6. The *idTracker* column is a regular idTracker output, *idTracker I* is an output with provided interpolation. FERDA was provided with expert annotations. There were 1 for Sowbug-3 sequence, 48 for Ants-1, 30 for Ants-3 and 6 for Zebrafish.

7.3 Results Discussion

To understand the results, we need to explain the differences between the observed systems. *IdTracker* is equipped with an approach which tries to solve interactions by applying morphological operations (erosion and dilation) on over-segmented regions. this leads to longer tracklets which in turn leads to higher accuracy due to the higher number of ID-Assigned regions.

IdTracker obtains the training set in a smart way using the idea of ID-Consistency rules. It finds all the places where animals are separated. This provides it with several sets of tracklets, where all IDs are present. Then it fixes one such set as the training set and tries to find an ID permutation for all others, and estimates the probability of such permutation. Then, it takes all sets matched with the training set with high probability and adds these tracklets to the training set.

The last significant difference is that *idTracker* is not using all regions for training and classification but it has a metric for picking good ones. That might be the reason for *FERDA*'s poor performance on Cam1 dataset, where the ants are climbing on the walls of Petri dish quite often, completely changing their appearance.

The presented results show that when *FERDA* is provided with a larger training set, it can outperform *idTracker*.

Chapter 8

Implementation Details

Although the system works on various species, the primary focus on ants also affected the name pick. Ferda is a main ant character in Ondřej Sekora's fairy tales from the world of insect. These fairy tales were quite popular during an author's childhood in the Czech Republic. Apart from that, it is an acronym for Fast Extremal Region Detector of Ants.

During the FERDA system development, we have been cooperating and discussing a lot with Barbara Casillas-Perez (Cremer Group) from Institute of Science and Technology Austria. Their main focus is on individual and colony-level antipathogen defenses in ants. This fruitful partnership gave us an opportunity to see the problem from the side of a target user. This led us to develop many tools to allow easier algorithm analysis, debugging and even data storing.

Along with our collaboration partners we have developed utilities to parallelize the most computationally demanding *FERDA* tasks in a cluster running gridEngine, as well integration with matlab post-processing pipeline, compatible with *Ctrax* output (which allow for further *JAABA* analysis).

Language	files	blank	comment	code
Python	312	13503	9083	106557
C++	3	194	447	2711
XML	7	0	0	1691
TEX	28	4	3	668
Cython	2	37	13	86
Markdown	3	14	0	72
Bourne Shell	1	7	4	34
C/C++ Header	1	5	2	21
SUM	357	13764	9552	111840

Table 8.1: A statistics of *FERDA* project showing the comprehensiveness of this work. It describes the number of lines except for the files column, describing number of files in given languge.

The method implementation and most of the supporting codes and experiments were implemented in *Python 2.7*. Several great libraries were used: *OpenCV* [29] and *Scikit-Image* [30] for image processing; *scikit-learn* [30] for

machine learning algorithms; *PuLP* [31] for Linear Programming optimisation; *NumPy* [32] as an linear algebra toolkit; *Matplotlib* [33] for plotting and visualisations; *graph-tool* [34] for efficient graph representation and manipulations and *PyQt* [35] the python wrapper for *Qt* [36] library for graphical user interface implementation.

The software is compatible with all desktop platforms (Linux, Mac OS X, Windows).

The method and the implementation were designed with an emphasis on modularity and generality which makes it easier to tune particular parts or to replace them with more specialized ones based on the problem domain it is used for.

8.1 Video Compression

One of the problems we have faced was the space consumption of many hours of video sequences capturing the experiments. There is a compromise between space consumption and the video quality. We have designed a solution which processes the video and creates two video files. First one is highly compressed and serves as a backup. The quality of an original video is preserved in the second file. The size reduction is achieved by replacing non-interesting parts of the frame with constant color (e.g. white or black, depending on the color of objects). When a sequence is modified in this way, current video codecs do the rest of the work. Our system can handle single video input or a pair of videos. In the second case, the frame is reconstructed by replacing the constant background with pixels from the low-quality video. In practice, this approach achieves size reduction by a factor of 10-15 while maintaining the same video quality. This method is applicable even for real-time video compression during an experiment recording.

8.2 Graph Viewer

Graph Viewer is one of the key tools for viewing a current state of a graph. It is useful not only for a programmer, but also for a user. A sophisticated algorithm for positioning of nodes (regions) provides neat and concise overview of the situation in the whole sequence and fast rendering libraries make it possible to change the visualisation in negligible time.

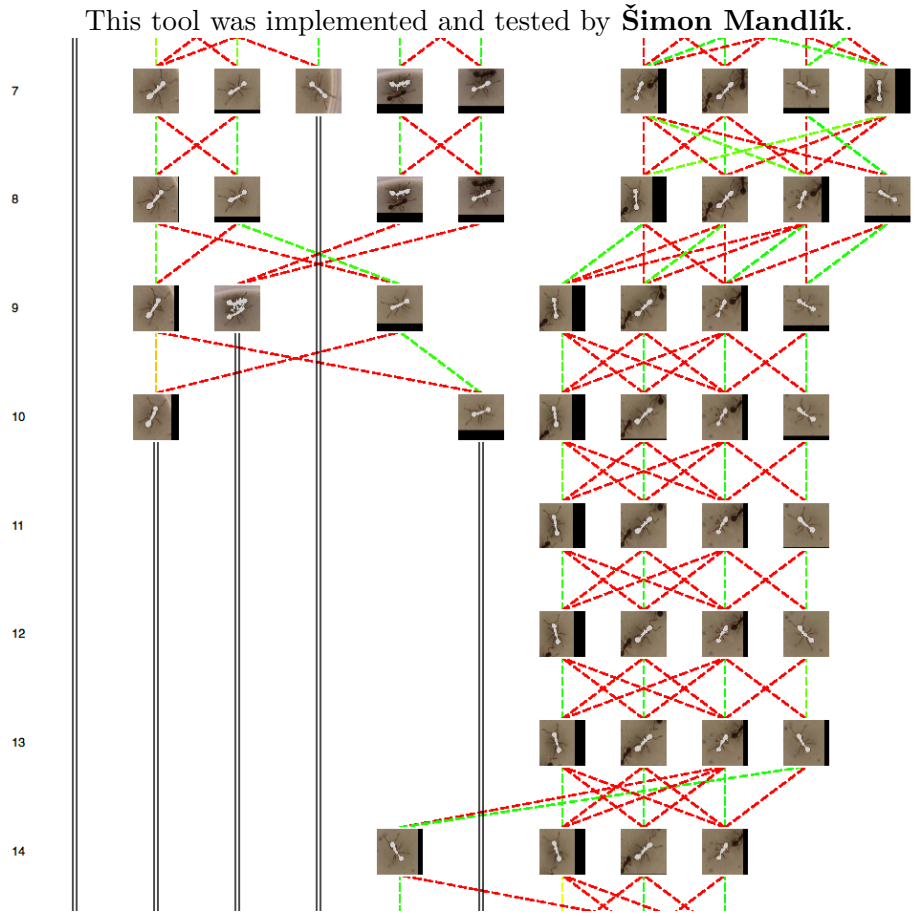


Figure 8.1: A screenshot from FERDA-graph-viewer tool implemented by Šimon Mandlík depicting a progress of tracklet creation. The edge color visualizes the likelihood of full ID-Set transfer starting with red (not likely) continues through yellow to green (very likely).)

8.3 Results Viewer

A results viewer is a complex tool allowing a user to go through the video sequence, visualize and even correct tracking results. It is used for expert annotations. It has a capability for position Ground Truth creation from scratch or on top of the results. It provides tools for comparison and visualization of other tracking systems. Search based on region or tracklet ids is possible. Thus it is very useful during the debugging process.

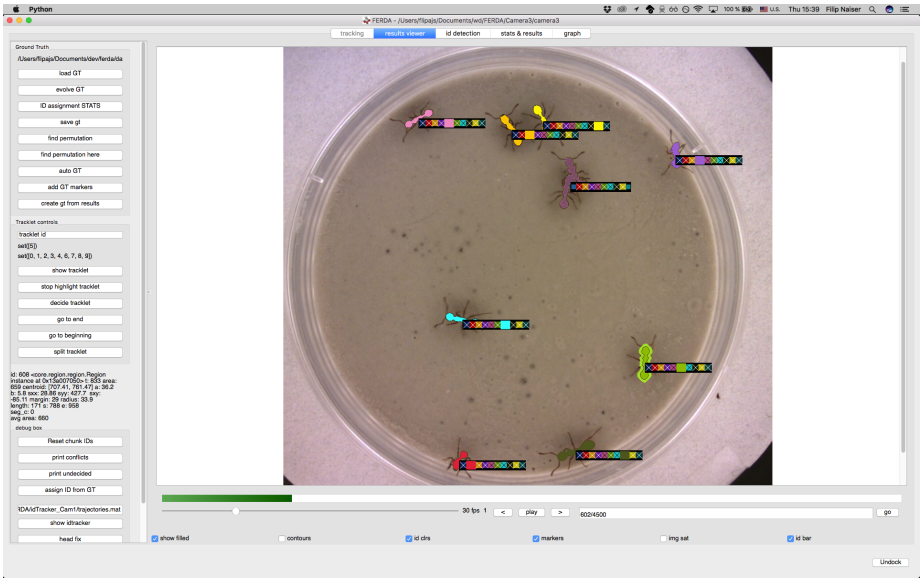


Figure 8.2: A screenshot from FERDA-results-viewer tool. In the middle part, there is a video player with results showed. P and N ID-Sets are visualized by stripes with squares. Where color describes an ID. When it is crossed, it is in N set, when enhanced it is in P set. When segmentation color is dotted it does not represent object ID but tracklet id. Belov is the video player and visualization control panel. In the left part, controls for GT creation, other system's results from visualisation and comparison and debugging tool are situated.

8.4 RFC Segmentation Tool

When preprocessing the images, we use random forest classifier (RFC) which requires following features to classify each pixel:

- Features 1 to 4 include basic color channels (RGB) and grayscale data
- Feature 5 detects edges using canny edge detector
- Features 6-8 attempt to highlight and eliminate delicate but undesirable shapes, such as antennas and legs, by subtracting neighboring pixels of a grayscale image.
- Feature 9 aims at the same goal, computing average grayscale value expected on each pixel based on neighboring pixel values.
- Features 10-12 compute differences of each pair of color channels
- Features 13-14 find the greatest/smallest neigbor value for each pixel in a grayscale image

Segmentation tool improves it's results by computing features on different scales using multiscale image pyramid. All of the features are computed on D images, where D is the depth of multiscale pyramid (number of downscales for the image). This helps when working with animals of various sizes and

h

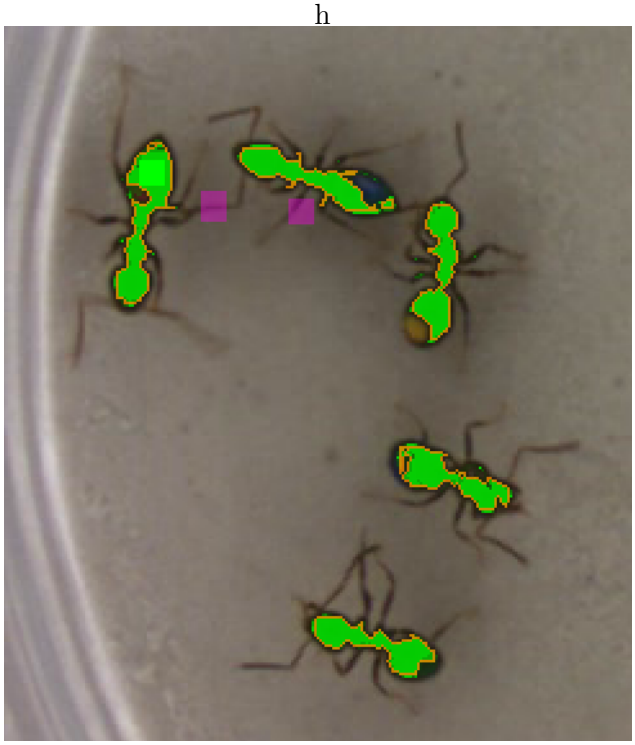


Figure 8.3: Example of RFC, per pixel classification outputs. Squares show user input marks, green represents foreground and purple represents background - legs and antennas are to be ignored in this case. Light green is the output probability map computed by RFC. Orange color shows resulting MSER contours, computed on probability map.

provides us with Dx14 features. The RFC classifier is then trained. Since computing large amount of features for each video frame would be time consuming, insignificant features (with RFC importance lower than certain threshold) are removed from the training set, the RFC is re-trained and reduced feature subset is used in subsequent computations. On Ants-1 and Ants-3 datasets, only 6-8 features on different scales proved useful to RFC, however, providing more features and scales can help when working with various datasets. The feature set is not final and might be further extended and modified to respond to new challenges. Visualisation of RFC segmentation output can be seen in Figure 8.3

This tool was implemented and tested by **Dita Hollmannová**.

Chapter 9
Conclusion

In this thesis, we have described a *Multi-Object Tracking* method for animal surveillance in biological experiments where the subjects are nearly indistinguishable by human eye. The method processes a video sequence and returns positions of subjects in each frame. On top of such output, a behavior analysis using specialized tools like *JAABA* can be performed. Using segmentation, the input is transformed into a set of regions represented by a graph with vertices representing regions and edges connecting regions in consecutive frames. Then the **K**-(almost)-disjoint path search is designed as a labelling problem.

The identity confusion during object interactions where occlusion occurs was addressed by *ID assignment* once the interaction ends. We have enhanced current state-of-the-art classifier for *ID assignment* and we have shown that it improves the performance for per region classification by 4.5%–18% (Table 7.1). Furthermore, we have supported the *ID assignment* with a set of consistency rules, which in most cases reduces the possible search space and improves the *ID assignment* reliability. We have also added an option for the user to provide an annotation when an *ID assignment* violates the *ID consistency rules (ICR)* or when the *ID assignment* certainty drops bellow given threshold.

The overall system performance was evaluated in three different settings on four different datasets (ants with a small color mark on the abdomen, ants without any tagging, sowbugs and zebrafish) and compared with the current state-of-the-art method *idTracker*.

The system development was discussed with Barbara Casillas-Perez, Cremer Group, Institute of Science and Technology Austria. Who is currently using our tool (without *ID detection*) for gathering global statistics from experiments on colony-level antipathogen defenses in ants and in these days, the testing phase of *ID assignment* on their experiments is running. For these purposes, a Graphical User Interface was implemented.

9.1 Future Work

The drawbacks of the current approach with *ICR* are in the hard decisions made. Its strength is a double-edged sword and once a wrong decision is made, it affects other tracklets. In future work we want to relax this by

expressing *ICR* with probabilities based on the tracklet lengths as the short tracklet classification are more prone to an error in *ID assignment*. The next thing we want to do is a deeper examination of *ID inconsistency* detections and automatic deduction of constraints.

Bibliography

[1] K. B. et al., "Ctrax the caltech multiple walking fly tracker." http://ctrax.sourceforge.net/, 2011.

[2] K. Branson, A. A. Robie, J. Bender, P. Perona, and M. H. Dickinson, "High-throughput ethomics in large groups of Drosophila," *Nature Methods*, vol. 6, pp. 451–457, June 2009.

[3] M. Kabra, A. A. Robie, M. Rivera-Alba, S. Branson, and K. Branson, "JAABA: interactive machine learning for automatic annotation of animal behavior," *Nature Methods*, vol. 10, pp. 64–67, Jan. 2013.

[4] A. Pérez-Escudero, J. Vicente-Page, R. C. Hinz, S. Arganda, and G. G. de Polavieja, "idTracker: tracking individuals in a group by automatic identification of unmarked animals," *Nature Methods*, vol. 11, pp. 743–748, July 2014.

[5] A. I. Dell, J. A. Bender, K. Branson, I. D. Couzin, G. G. de Polavieja, L. P. J. J. Noldus, A. Pérez-Escudero, P. Perona, A. D. Straw, M. Wikelski, and U. Brose, "Automated image-based tracking and its application in ecology," *Trends in Ecology & Evolution*, vol. 29, pp. 417–428, July 2014.

[6] M. Henry, M. Béguin, F. Requier, O. Rollin, J.-F. Odoux, P. Aupinel, J. Aptel, S. Tchamitchian, and A. Decourtye, "A Common Pesticide Decreases Foraging Success and Survival in Honey Bees," *Science*, vol. 336, pp. 348–350, Apr. 2012.

[7] C. W. Schneider, J. Tautz, B. Grünewald, and S. Fuchs, "RFID Tracking of Sublethal Effects of Two Neonicotinoid Insecticides on the Foraging Behavior of Apis mellifera," *PLOS ONE*, vol. 7, p. e30023, Jan. 2012.

[8] M. E. O'Neal, D. A. Landis, E. Rothwell, L. Kempel, and D. Reinhard, "Tracking Insects with Harmonic Radar: a Case Study," *American Entomologist*, vol. 50, pp. 212–218, Oct. 2004.

[9] D. Psychoudakis, W. Moulder, C. C. Chen, H. Zhu, and J. L. Volakis, "A Portable Low-Power Harmonic Radar System and Conformal Tag for Insect Tracking," *IEEE Antennas and Wireless Propagation Letters*, vol. 7, pp. 444–447, 2008.

[10] T. Kimura, M. Ohashi, K. Crailsheim, T. Schmickl, R. Okada, G. Radspieler, and H. Ikeno, "Development of a New Method to Track Multiple Honey Bees with Complex Behaviors on a Flat Laboratory Arena," *PLOS ONE*, vol. 9, p. e84656, Jan. 2014.

[11] T. Kimura, M. Ohashi, K. Crailsheim, T. Schmickl, R. Okada, G. Radspieler, and H. Ikeno, "K-track application." http://software.incf.org/software/the-tracking-software-k-track, 2014.

[12] N. A. Swierczek, A. C. Giles, C. H. Rankin, and R. A. Kerr, "High-throughput behavioral analysis in C. elegans," *Nature Methods*, vol. 8, pp. 592–598, July 2011.

[13] A. Pérez-Escudero, "idtracker software." http://www.idtracker.es, 2014.

[14] F. Naiser, "Detection, description and tracking of ants in video sequences." http://cyberold.felk.cvut.cz/research/theses/papers/536.pdf, 2014.

[15] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, pp. 761–767, Sept. 2004.

[16] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001.

[17] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-Based Anomaly Detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, pp. 3:1–3:39, Mar. 2012.

[18] "Isolation forest, scikit-learn documentation.." http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html, 2017.

[19] D. W. H. Jr and S. Lemeshow, *Applied Logistic Regression*. John Wiley & Sons, Oct. 2004. Google-Books-ID: Po0RLQ7USIMC.

[20] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, Jan. 1967.

[21] D. Arthur and S. Vassilvitskii, "K-means++: The Advantages of Careful Seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, (Philadelphia, PA, USA), pp. 1027–1035, Society for Industrial and Applied Mathematics, 2007.

[22] "K-means++ article on wikipedia.org." https://en.wikipedia.org/wiki/K-means

[23] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, pp. 179–187, Feb. 1962.

[24] J. v. d. Weijer, C. Schmid, J. Verbeek, and D. Larlus, "Learning Color Names for Real-World Applications," *IEEE Transactions on Image Processing*, vol. 18, pp. 1512–1523, July 2009.

[25] L. Zhang, R. Chu, S. Xiang, S. Liao, and S. Z. Li, "Face Detection Based on Multi-Block LBP Representation," in *Advances in Biometrics*, pp. 11–18, Springer, Berlin, Heidelberg, Aug. 2007. DOI: 10.1007/978-3-540-74549-5_2.

[26] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, June 2005.

[27] A. Pérez-Escudero, "A video sequence with five zebrafish in a shallow aquarium.." http://www.idtracker.es/download, 2014.

[28] "A video sequence with five sowbugs in a petri dish.." https://youtu.be/FLrNHSS59tY, 2011.

[29] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[31] S. Mitchell, M. OSullivan, and I. Dunning, "Pulp: a linear programming toolkit for python," *The University of Auckland, Auckland, New Zealand, http://www. optimization-online. org/DB_FILE/2011/09/3178. pdf*, 2011.

[32] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[33] J. D. Hunter *et al.*, "Matplotlib: A 2d graphics environment," *Computing in science and engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[34] T. P. Peixoto, "The graph-tool python library," *figshare*, 2014.

[35] Riverabank, "Pyqt4." https://riverbankcomputing.com/software/pyqt.

[36] Qt, "Qt4." https://www.qt.io.

Appendix A

RFC Tuning Results

The results of Random Forest Classifier tuning as described insection 5.3

A.0.1 Max Depth of a Decision Tree

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-4.15%	-1.61%	-3.03%	-2.73%
Colornames	-2.02%	**+0.02%**	-6.74%	-3.44%
C-co-occurence	-0.17%	-0.93%	-1.21%	-1.67%
I-co-occurence	-0.26%	-5.57%	-3.52%	-4.57%
LBP	-2.56%	-0.22%	-0.81%	-2.05%
HoG	-2.02%	-0.22%	-2.16%	-2.60%
An average	-1.86%	-1.42%	-2.91%	-2.84%

Table A.1: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-d** = 5. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-0.20%	**+0.52%**	**+0.09%**	**+0.18%**
Colornames	-0.17%	**+0.08%**	**+0.45%**	-0.03%
C-co-occurence	0.00%	**+0.01%**	0.00%	**+0.12%**
I-co-occurence	0.00%	-0.03%	0.00%	**+0.12%**
LBP	**+0.26%**	**+0.29%**	-0.23%	**+0.06%**
HoG	**+0.18%**	**+0.41%**	**+0.08%**	-0.17%
An average	**+0.01%**	**+0.21%**	**+0.07%**	**+0.05%**

Table A.2: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-d** = 10. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	0.00%	**+0.15%**	0.00%	-0.01%
Colornames	0.00%	**+0.03%**	**+0.12%**	-0.07%
C-co-occurence	0.00%	0.00%	0.00%	0.00%
I-co-occurence	0.00%	0.00%	0.00%	0.00%
LBP	0.00%	-0.03%	**+0.01%**	0.00%
HoG	0.00%	**+0.03%**	0.00%	**+0.01%**
An average	0.00%	**+0.03%**	**+0.02%**	-0.01%

Table A.3: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-d** = 15. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

A.1 Min Samples In a Leaf

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-0.13%	**+0.45%**	-0.15%	-0.51%
Colornames	-0.08%	**+0.16%**	**+0.61%**	-0.43%
C-co-occurence	-0.13%	**+0.31%**	**+0.28%**	**+0.55%**
I-co-occurence	-0.09%	-0.14%	-0.15%	**+0.06%**
LBP	**+1.13%**	**+0.58%**	**+0.65%**	**+0.27%**
HoG	**+0.95%**	**+0.47%**	**+0.60%**	-0.04%
An average	**+0.28%**	**+0.30%**	**+0.31%**	-0.01%

Table A.4: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **min-leaf** = 2 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-0.48%	**+0.84%**	-0.30%	-0.75%
Colornames	-0.87%	**+0.41%**	**+0.46%**	-0.32%
C-co-occurence	-0.26%	**+0.58%**	**+0.05%**	**+0.18%**
I-co-occurence	-0.08%	-0.13%	-0.14%	**+0.06%**
LBP	**+1.06%**	**+0.98%**	**+0.77%**	**+0.86%**
HoG	**+1.29%**	**+0.77%**	**+0.57%**	**+0.23%**
An average	**+0.11%**	**+0.58%**	**+0.24%**	**+0.04%**

Table A.5: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **min-leaf** = 3. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-2.42%	**+1.09%**	-1.16%	-2.07%
Colornames	-1.24%	**+0.54%**	-0.50%	-1.40%
C-co-occurence	-0.59%	**+0.61%**	-0.13%	**+0.25%**
I-co-occurence	-0.46%	-0.02%	-1.02%	-0.21%
LBP	-0.36%	**+1.03%**	**+1.26%**	**+0.59%**
HoG	-0.07%	**+0.99%**	-0.13%	-0.35%
An average	-0.86%	**+0.71%**	-0.28%	-0.53%

Table A.6: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **min-leaf** = 5 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

A.1.1 Max Features

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-3.62%	-2.27%	-6.73%	-2.49%
Colornames	-3.83%	-0.93%	-1.75%	-0.34%
C-co-occurence	**+0.76%**	-0.10%	**+0.04%**	**+1.46%**
I-co-occurence	**+0.80%**	**+0.42%**	**+0.61%**	**+0.84%**
LBP	-1.90%	-1.02%	-2.50%	-1.58%
HoG	-1.73%	-1.17%	-1.49%	-1.46%
An average	-1.59%	-0.84%	-1.97%	-0.59%

Table A.7: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 10 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-1.38%	-0.77%	-0.61%	-0.53%
Colornames	-1.88%	-0.32%	-0.01%	-0.29%
C-co-occurence	**+0.44%**	-0.07%	**+0.33%**	**+0.52%**
I-co-occurence	**+0.48%**	-0.15%	**+0.44%**	**+0.54%**
LBP	-0.37%	-0.51%	-1.29%	-0.69%
HoG	-0.66%	-0.36%	-0.88%	-0.40%
An average	-0.56%	-0.36%	-0.34%	-0.14%

Table A.8: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 20 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-0.60%	-0.25%	-0.14%	-0.37%
Colornames	**+0.10%**	-0.20%	**+0.35%**	**+0.10%**
C-co-occurence	**+0.50%**	**+0.18%**	-0.04%	**+0.26%**
I-co-occurence	**+0.29%**	**+0.17%**	**+0.30%**	**+0.22%**
LBP	-0.25%	**+0.05%**	-1.17%	-0.53%
HoG	-0.32%	-0.06%	**+0.13%**	-0.22%
An average	-0.05%	-0.02%	-0.09%	-0.09%

Table A.9: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 30 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-0.36%	**+0.15%**	**+0.13%**	-0.07%
Colornames	-0.13%	-0.10%	**+0.34%**	**+0.10%**
C-co-occurence	**+0.04%**	-0.01%	**+0.07%**	**+0.26%**
I-co-occurence	-0.16%	-0.04%	-0.10%	**+0.27%**
LBP	**+0.43%**	-0.15%	-0.78%	-0.43%
HoG	**+0.02%**	**+0.05%**	-0.21%	-0.33%
An average	-0.03%	-0.02%	-0.09%	-0.03%

Table A.10: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 40. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	**+0.07%**	**+0.11%**	**+0.41%**	-0.40%
Colornames	-0.26%	-0.29%	**+0.14%**	-0.04%
C-co-occurence	**+0.06%**	-0.00%	-0.09%	-0.05%
I-co-occurence	-0.14%	-0.18%	-0.24%	-0.07%
LBP	**+0.23%**	**+0.28%**	-0.05%	-0.09%
HoG	**+0.41%**	**+0.30%**	**+0.39%**	**+0.01%**
An average	**+0.06%**	**+0.04%**	**+0.09%**	-0.11%

Table A.11: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 60. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	**+0.01%**	**+0.21%**	**+0.25%**	-0.65%
Colornames	-0.05%	-0.12%	-0.01%	-0.65%
C-co-occurence	-0.10%	-0.07%	-0.44%	-0.11%
I-co-occurence	-0.38%	-0.65%	-0.32%	**+0.15%**
LBP	**+0.36%**	**+0.17%**	-0.26%	-0.01%
HoG	**+0.37%**	**+0.02%**	**+0.33%**	-0.50%
An average	**+0.04%**	-0.07%	-0.07%	-0.30%

Table A.12: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 70 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	**+0.07%**	**+0.24%**	**+0.07%**	-0.50%
Colornames	**+0.28%**	-0.29%	-0.31%	-0.56%
C-co-occurence	-0.27%	-0.13%	-0.86%	-0.73%
I-co-occurence	-0.41%	-0.46%	-0.72%	-0.48%
LBP	**+1.03%**	**+0.09%**	-0.44%	-0.17%
HoG	**+0.31%**	**+0.24%**	**+0.23%**	**+0.03%**
An average	**+0.17%**	-0.05%	-0.34%	-0.40%

Table A.13: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 80 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-0.61%	**+0.34%**	**+0.00%**	-0.69%
Colornames	-0.06%	-0.52%	-0.98%	-1.03%
C-co-occurence	-0.40%	-0.23%	-0.58%	-0.88%
I-co-occurence	-0.55%	-0.55%	-0.77%	-0.17%
LBP	**+1.10%**	**+0.25%**	-0.02%	**+0.19%**
HoG	**+0.52%**	**+0.29%**	-0.02%	-0.61%
An average	**+0.00%**	-0.07%	-0.39%	-0.53%

Table A.14: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** = 100 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	-1.38%	-0.77%	-0.33%	-0.51%
Colornames	-4.24%	-1.39%	-3.25%	-0.95%
C-co-occurence	**+0.54%**	-2.77%	-1.09%	**+1.22%**
I-co-occurence	**+1.09%**	-1.05%	**+0.44%**	**+0.55%**
LBP	-1.06%	-0.54%	-3.74%	-2.45%
HoG	-1.36%	-0.53%	-0.65%	-0.77%
An average	-1.07%	-1.17%	-1.44%	-0.49%

Table A.15: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **max-f** $= \sqrt{f}$, where f is the numbef of features . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

A.1.2 Number of Decision Trees

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	**+1.61%**	**+1.46%**	**+2.00%**	**+1.26%**
Colornames	**+2.49%**	**+1.51%**	**+4.52%**	**+1.61%**
C-co-occurence	**+0.96%**	**+2.61%**	**+3.01%**	**+2.66%**
I-co-occurence	**+1.16%**	**+2.47%**	**+3.04%**	**+1.96%**
LBP	**+5.69%**	**+2.57%**	**+3.91%**	**+5.08%**
HoG	**+4.44%**	**+2.85%**	**+4.34%**	**+3.11%**
An average	**+2.72%**	**+2.24%**	**+3.47%**	**+2.61%**

Table A.16: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** $= 20$. Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	**+1.87%**	**+1.98%**	**+2.40%**	**+1.39%**
Colornames	**+2.84%**	**+2.09%**	**+5.89%**	**+2.70%**
C-co-occurence	**+1.17%**	**+3.73%**	**+3.75%**	**+3.48%**
I-co-occurence	**+1.38%**	**+3.49%**	**+4.56%**	**+2.90%**
LBP	**+6.98%**	**+4.00%**	**+6.59%**	**+7.30%**
HoG	**+6.49%**	**+3.96%**	**+6.26%**	**+5.33%**
An average	**+3.45%**	**+3.21%**	**+4.91%**	**+3.85%**

Table A.17: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** $= 30$. Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	+1.91%	+2.35%	+2.67%	+1.78%
Colornames	+2.87%	+2.24%	+6.62%	+2.71%
C-co-occurence	+1.67%	+3.91%	+4.68%	+3.77%
I-co-occurence	+1.52%	+3.91%	+5.15%	+3.38%
LBP	+8.94%	+4.82%	+8.28%	+8.94%
HoG	+8.05%	+4.83%	+7.34%	+5.89%
An average	+4.16%	+3.68%	+5.79%	+4.41%

Table A.18: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 40 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	+2.31%	+2.48%	+3.05%	+2.14%
Colornames	+3.53%	+2.55%	+7.36%	+3.05%
C-co-occurence	+1.62%	+4.23%	+5.11%	+4.06%
I-co-occurence	+1.65%	+3.99%	+5.35%	+3.29%
LBP	+8.97%	+5.18%	+9.49%	+9.44%
HoG	+9.09%	+5.41%	+8.23%	+6.20%
An average	+4.53%	+3.97%	+6.43%	+4.70%

Table A.19: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 50 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	+2.20%	+2.64%	+3.14%	+2.04%
Colornames	+3.77%	+2.77%	+7.90%	+3.47%
C-co-occurence	+1.75%	+4.59%	+5.37%	+4.60%
I-co-occurence	+1.74%	+4.74%	+6.08%	+3.93%
LBP	+10.34%	+6.12%	+11.00%	+10.53%
HoG	+10.07%	+6.03%	+9.04%	+6.90%
An average	+4.98%	+4.48%	+7.09%	+5.24%

Table A.20: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 75 . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	+2.63%	+2.85%	+3.47%	+2.37%
Colornames	+3.81%	+2.95%	+8.44%	+3.47%
C-co-occurence	+1.91%	+4.86%	+5.68%	+4.76%
I-co-occurence	+1.69%	+4.96%	+6.34%	+4.37%
LBP	+10.91%	+6.71%	+12.67%	+11.04%
HoG	+10.54%	+6.55%	+10.05%	+7.39%
An average	+5.25%	+4.81%	+7.77%	+5.57%

Table A.21: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 100. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

	Ants-1	Zebrafish-1	Ants-3	Sowbug-3
Moments	+2.80%	+2.98%	+3.57%	+2.44%
Colornames	+4.17%	+3.11%	+9.05%	+3.60%
C-co-occurence	+1.94%	+5.10%	+6.04%	+5.35%
I-co-occurence	+1.77%	+5.25%	+6.73%	+4.64%
LBP	+11.88%	+7.16%	+14.36%	+12.06%
HoG	+11.84%	+7.26%	+11.09%	+7.82%
An average	+5.73%	+5.14%	+8.47%	+5.98%

Table A.22: The change in Random Forest Classifier performance w.r.t. the default setting in Table 5.1. Parameter change: **n-trees** = 200. . Dataset names are in columns, feature spaces are in rows. Last row shows an average of changes.

Appendix B

Tracker Comparison Diagrams

Figure B.1: ID detection results on Sowbug-3, Ants-1, Ants-3, Zebrafish-1 datasets. Y axis labels identities, X axis labels frames. Top row - *idTracker* (**with** interpolation), middle (thinner) row - ground truth, bottom row - FERDA **(without ID-Consistency Rules and without expert annotations)**. Gray - ID unassigned.

Figure B.2: ID detection results on Sowbug-3, Ants-1, Ants-3, Zebrafish-1 datasets. Y axis labels identities, X axis labels frames. Top row - idTracker (**without** interpolation), middle (thinner) row - ground truth, bottom row - FERDA **(without ID-Consistency Rules and without expert annotations)**. Gray - ID unassigned.

Figure B.3: ID detection results on Sowbug-3, Ants-1, Ants-3, Zebrafish-1 datasets. Y axis labels identities, X axis labels frames. Top row - *idTracker* (**with** interpolation), middle (thinner) row - ground truth, bottom row - FERDA **(without expert annotations)**. Gray - ID unassigned.

Figure B.4: ID detection results on Sowbug-3, Ants-1, Ants-3, Zebrafish-1 datasets. Y axis labels identities, X axis labels frames. Top row - idTracker (**without** interpolation), middle (thinner) row - ground truth, bottom row - FERDA **(without expert annotations)**. Gray - ID unassigned.