Institut für Informatik
Biorobotics Lab

Portierung und Weiterentwicklung von Trackingalgorithmen in das BioTracker Framework

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science
(B.Sc.)
im Fach Informatik

von
Patrick Winterstein
geboren am 24.08.1992 in Berlin

Betreuung:

1. *Prof. Dr. Tim Landgraf*
2. *Prof. Dr. Raúl Rojas*

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Bachelorarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Berlin, den 21.07.2017

Zusammenfassung

Am Biorobotics Lab wird Forschung zu kollektiven Systemen betrieben. Hierzu werden z.B. Honigbienenkolonien oder Fischschwärme gefilmt und das Verhalten der Individuen und des Kollektivs analysiert. Als Grundlage der Verhaltensanalyse dienen Bewegungsspuren, die mittels Computervision aus Videoaufnahmen extrahiert werden.

Für jedes Modellsystem wurden hierfür jeweils eigenständige Programme entwickelt. Um diese Programme zu zentralisieren wurde der BioTracker entwickelt, ein Computervision Framework welches Grundfunktionalitäten für das Tracking zur Verfügung stellt. Dieses ist durch so genannte Trackingmodule erweiterbar, welche die verschiedenen Trackingalgorithmen kapseln.

Der Fokus dieser Arbeit liegt auf dem Portieren von zwei verschiedenen Algorithmen in Trackingmodule für den BioTracker und der Analyse der Qualität der gelieferten Ergebnisse.

Danksagung

Die Fertigstellung der Arbeit wäre ohne die Unterstützung einiger Menschen nicht möglich gewesen und ich möchte mich hier bei ihnen bedanken.

Zuerst möchte ich Prof. Dr. Tim Landgraf für die Betreuung dieser Arbeit danken. Er hatte immer ein offenes Ohr, wenn Schwierigkeiten auftraten oder mir die Herangehensweise an eine Problemstellung nicht klar war. Danke.
Des Weiteren möchte ich Prof. Dr. Raúl Rojas für die Übernahme der Rolle des Zweitgutachters danken.
Außerdem möchte ich meinen Dank für die Unterstützung der anderen Mitglieder des Biorobotics Labs aussprechen, die immer bei Fragen zu den alten Trackingprogrammen zur Verfügung standen.

Danken möchte ich auch meinen besten Freunden Robert Kieschke und Marcel Miszalski, die immer für Ablenkung gesorgt haben, wenn es gerade nötig war oder einfach nur bei Schwierigkeiten zugehört haben, obwohl sie nicht immer alles verstehen konnten.

Zu guter Letzt möchte ich meinen Eltern Sabine und Heinz Winterstein, sowie dem Rest meiner Familie, für die Unterstützung und die Geduld während meines Studiums danken.

Danke.

Inhaltsverzeichnis

1 Einführung **1**
- 1.1 Biorobotics Lab . . . 1
- 1.2 Motivation des BioTrackers . . . 1
- 1.3 BioTracker Framework . . . 2
- 1.4 Trackingalgorithmen . . . 2

2 State of the Art **4**
- 2.1 Externe Trackingprogramme . . . 4
  - 2.1.1 Trackingframeworks . . . 4
  - 2.1.2 Ctrax . . . 4
  - 2.1.3 idtracker . . . 5
- 2.2 Interne Trackingprogramme . . . 6
  - 2.2.1 BeeDance Tracker . . . 6
  - 2.2.2 Fishtracker . . . 7

3 Implementierung **8**
- 3.1 Anforderungen . . . 8
  - 3.1.1 RigidFlow Tracker . . . 8
  - 3.1.2 SimpleTracker . . . 8
- 3.2 Beschreibung der Trackingalgorithmen . . . 8
  - 3.2.1 RigidFlow Tracker . . . 8
  - 3.2.2 SimpleTracker . . . 11

4 Evaluierung **15**
- 4.1 Benutzbarkeit . . . 15
- 4.2 Zuverlässigkeit . . . 15
  - 4.2.1 RigidFlow Tracker . . . 16
  - 4.2.2 SimpleTracker . . . 19

5 Diskussion **23**
- 5.1 RigidFlow Tracker . . . 23
- 5.2 SimpleTracker . . . 24
- 5.3 Parameterauswahl . . . 25

6 Ausblick **26**
- 6.1 BioTracker Framework . . . 26
- 6.2 RigidFlow Tracker . . . 26
- 6.3 SimpleTracker . . . 27

6.4 Veröffentlichung BioTracker 27

A Anhang **28**
A.1 Datenanalyse 28

Literatur **32**

Abbildungsverzeichnis

3.1 Screenshot RigidFlow Tracker 11
3.2 Screenshot SimpleTracker 13

4.1 Beispielbild des Videos für den RigidFlow Tracker 16
4.2 Positionabstände beim RigidFlow Tracker (10/17/1000) 17
4.3 Ausrichtungsdifferenzen beim RigidFlow Tracker (10/17/1000) 18
4.4 Beispielbild des Videos für den SimpleTracker 19
4.5 Fehlende Detektionen nach Pfaden 20
4.6 Vergleich der Pfade in den manuell und automatisch getrackten Daten 21

5.1 Abstände beim Tracking mit dem SimpleTracker in 10er Schritten 25

Kapitel 1

1 Einführung

1.1 Biorobotics Lab

Das Biorobotics Lab ist Teil der Arbeitsgruppe 'Artificial & Collective Intelligence', welche am Institut für Informatik an der Freien Universität beheimatet ist. Die Arbeitsgruppe ist außerdem Mitglied in dem 'Dahlem Center for Machine Learning and Robotics', einer Kooperation mehrerer Arbeitsgruppen der Freien Universität, welche intelligente Systeme und künstliche Intelligenz erforscht. Im Biorobotics Lab wird Grundlagenforschung zu kollektiver Intelligenz betrieben. Hierfür wird das Verhalten von Tieren, vor allem von Bienen, Hummeln und Fischen, erforscht, wobei unter anderem Roboter in die Kollektive der Tiere eingeschleust werden. Diese Roboter werden einerseits zur Beobachtung der Tiere genutzt, andererseits können mit ihnen aber auch versucht werden, das Kollektiv zu beeinflussen. Die gewonnenen Erkentnisse könnten in vielen anderen Forschungsfelder Anwendungen finden, bspw. in Schwärmen von autonomen Fahrzeugen [1].

1.2 Motivation des BioTrackers

Um das Kollektiv verstehen zu können, muss das Verhalten der Individuen und deren Zusammenspiel erst einmal analysiert werden. Hierfür werden im Biorobotics Lab die Tiere typischerweise bei ihrem natürlichen Verhalten gefilmt und die daraus entstehenden Videos mit Computervision Methoden ausgewertet. Wobei zuerst die Tiere im Video verfolgt und die daraus resultierenden Daten, zur späteren Analyse auf verschiedenste Verhalten, gespeichert werden.

Im Biorobotics Lab wurden hierfür viele alleinstehende Trackingprogramme geschrieben, welche alle eine gewisse Grundfunktionalität liefern. Hierbei handelt es sich vor allem um das Laden und Abspielen der Videos, bzw. auch von Kamerastreams, sowie das Speichern und Laden der Trackingdaten. Um diese Grundfunktionalitäten nicht für jede neue Trackingmethode erneut Programmieren zu müssen, wurde das BioTracker Framework entwickelt. Die Auswertung der Ergebnisse fällt durch die Nutzung des Frameworks außerdem leichter, denn die Struktur der Daten ähnelt sich auch bei verschiedenen Trackern stark. Ein weiterer Vorteil eines zentralen Frameworks ist, dass Nutzer der Trackingalgorithmen nur noch ein Programm installieren

müssen, in welchem Sie dann verschiedenste Module für die Trackingalgorithmen laden können.

1.3 BioTracker Framework

Beim BioTracker handelt es sich um ein OpenSource C++ Framework für Computervision Anwendungen, welches auf allen großen Plattformen genutzt werden kann. Es stellt grundlegende Funktionalitäten zur Verfügung, welche in vielen Computervision Anwendungen genutzt werden. Hierzu zählen das Laden von Videos bzw. Kamerastreams, die Möglichkeit Trackingdaten zu speichern und zu laden, sowie eine grafische Oberfläche, die Interaktionen mit dem Video ermöglicht. Die grafische Oberfläche (GUI) ist von der Kernfunktionalität getrennt, wodurch die Nutzung des BioTrackers bspw. auf einem Supercomputer zur automatischen Analyse großer Datenmengen ermöglicht wird.

Der BioTracker stellt einige grundlegende Trackingalgorithmen zur Verfügung; es besteht aber auch die Möglichkeit den BioTracker mit eigenen Trackingalgorithmen zu erweitern. Hierfür muss ein sogenannter 'Tracker' implementiert werden. Dieser muss den Computervision-Algorithmus beinhalten, mit dem Objekte getrackt werden sollen. Des Weiteren besteht die Möglichkeit eigene GUI-Elemente zur Verfügung zu stellen, die bspw. zum Einstellen von Parametern, die für den Algorithmus benötigt werden, genutzt werden können. Außerdem kann das angezeigte Bild manipuliert werden. Dies wird vorallem zur Markierung der erkannten Objekte genutzt. Es besteht auch die Möglichkeit, Tracker außer in C++ in Python zu implementieren [2].

1.4 Trackingalgorithmen

Die in dieser Arbeit behandelten Trackingalgorithmen sind ursprünglich für Projekte des Biorobtics Lab entwickelt worden. Im ersten Projekt wird der Bienentanz erforscht; ein Kommunikationsmittel, mit dem Honigbienen bspw. Informationen über Futterstellen kommunizieren. Dafür wird eine Wabe eines Bienenstocks gefilmt und die daraus entstandenen Videos auf Tänze untersucht. Im Tracker soll nun die tanzende Biene verfolgt und damit die Position und die Ausrichtung festgestellt werden. Um dies in den gegebenen Videos umzusetzen, werden Merkmale der tanzenden Biene extrahiert und diese im Video verfolgt. Die Merkmale sind von einem Rechteck umschlossen, sodass die Position und die Ausrichtung berechnet werden können.

Der zweite Algorithmus wird für mehrere Projekte genutzt. In einem wird das Verhalten von Hummeln bei der Futtersuche analysiert, wobei eine Box mit verschiedenen möglichen Futterstellen von oben gefilmt wird. Die Hummeln können durch einen Eingang in die Box gelangen und in dieser nach Futter suchen, dabei ist immer nur eine der vier möglichen Futterstellen mit Futter bestückt. Aus den Videos sollen nun die Positionen und Ausrichtungen der Hummeln extrahiert werden.

Ein weiteres Projekt untersucht das Verhalten von Fischschwärmen. Hierbei wird ein Tank, indem sich ein Schwarm mit Fischen aufhält, von oben gefilmt. Auch hier sollen die Positionen

und Ausrichtungen extrahiert werden. Die entstehenden Videos ähneln sich bei beiden Projekten in bestimmten Punkten. Der Hintergrund ist sehr gut von der Tieren zu unterscheiden, da dieser sehr hell und die Tiere eher dunkel sind. Damit können die Konturen der Tiere durch Subtraktion des Hintergrunds aus jedem Bild extrahiert und somit die Position und Ausrichtung bestimmt werden.

Ziel dieser Arbeit ist es, beide Algorithmen in eigene Tracker für das BioTracker Framework zu übertragen. Soweit möglich werden die Tracker verallgemeinert, sodass diese auch für ähnliche Fälle nutzbar sind. Danach wird analysiert, wie es um die Qualität der Ergebnisse des Trackings für die Daten einiger der angesprochenen Projekte steht.

Kapitel 2

2 State of the Art

Das Kapitel 'State of the Art' behandelt den aktuellen Stand der Forschung bei Trackingprogrammen und zeigt sowohl Stärken und Schwächen ausgwählter externer (Ctrax 2.1.2 [3] [4], idtracker 2.1.3 [5]), sowie interner Tracker (Beedance Tracker 2.2.1, Fishtracker 2.2.2) auf. Im Weiteren werden zur Zeit benötigte Features für Projekte des Biorobotics Labs angesprochen.

2.1 Externe Trackingprogramme

2.1.1 Trackingframeworks

Die Anzahl an existierenden Trackingframeworks ist sehr gering und bereits Existierende weisen oft schwerwiegende Nachteile bei der Benutzung auf. Ein bereits vorhandenes OpenSource Trackingframework ist Swistrack, welches in C++ geschrieben ist. Hierbei wird das Tracking durch die Auswahl von Komponenten in den verschiedenen Stufen des Trackings (Bildvorverarbeitung, Objektfindung, Pfadzuwesiung, ...) bestimmt. Diese Komponenten sind durch Plugins erweiterbar. Eine sinnvolle Kombination an Komponenten auszuwählen ist für Nutzer, die sich nicht mit den technischen Details auskennen, fast unmöglich. Da die Trackingalgorithmen, die im Biorobotics Lab entwickelt werden, auch von Forschern aus anderen Fachgebieten, meist der Biologie, genutzt werden sollen, ist eine einfache Bedienung notwending und es wurde sich gegen eine Nutzung von Swistrack entschieden. Ein weiteres, bereits auf dem Markt verfügbares, Trackingframework ist ImageJ. Dieses bietet außerdem viele Funktionalitäten zur Manipulation von Bildern, die nicht fürs Tracking benötigt werden, wodurch es eine unübersichtliche Oberfläche aufweist. Die Tatsache, dass es auf Java basiert, schränkt die Geschwindigkeit des Frameworks ein. Da ImageJ's Oberfläche überladen und unintuitive ist und die Geschwindigkeit des Frameworks nicht den Ansprüchen im Biorobotics Lab entspricht, sollte auch dieses nicht genutzt werden.

2.1.2 Ctrax

Ctrax ist eine OpenSource Tracking Software zur Analyse von individuellem und sozialen Verhalten von Obstfliegen (Drosophila) und wurde am Howard Hughes Medical Institute in der Janelia-Gruppe entwickelt.

Die Software basiert größtenteils auf Matlab, einer Programmiersprache, die zur Auswertung großer Datenmengen in weitreichenden Fachbereichen, wie z.B. Biophysik, herangezogen wird.

Zur Vorverarbeitung des Videos wird 'Background Substraction' genutzt, dabei wird der Hintergrund von den einzelnen Frames des Videos abgezogen. Um die Objekte, die sich vom Hintergrund unterscheiden, herauszufiltern, werden eine untere und eine obere Schwelle festgelegt. Die obere Schwelle legt fest, ab wann ein Pixel als Vordergrund gewertet wird. Jeder Pixel, der um ein Vordergrundpixel liegt, wird auch als Vordergrund gewertet, solange er die untere Schwelle nicht unterschreitet. Der Hintergrund wird hierbei durch eine Menge an zufällig gewählten Bildern des Videos berechnet. Zur Reduzierung des benötigten Rechenaufwandes kann ein Bereich festgelegt werden, in dem das Tracking durchgeführt wird, wodurch Bereiche mit reinem Hintergrund ausgeschlossen werden können. Beim Tracking gefundene Objekte werden noch weiter verarbeitet, indem Objekte, deren Größe sich merklich vom Durchschnitt unterscheidet, getrennt, zusammengefasst oder verworfen werden. Im ersten Frame werden allen Objekten IDs zugeordnet. Diese werden anhand eines Bewegungsmodells, welches vorher festgelegt wurde, und den vorherigen Frames, den Objekten im berechneten Frame zugeordnet, sodass für jede ID ein Pfad entsteht. Hierbei wird angenommen, dass sich Objekte nicht aus dem Trackingbereich entfernen, weswegen Ctrax vor allem für 'closed world'-Setups geeignet ist.

Aufgrund der gewählten Herangehensweise bei der Vorverarbeitung ist Ctrax anfällig für Kompressionsartefakte, d.h. Stellen im Video, die aufgrund der Kompression der Bildsequenz, Unterschiede zum Originalbild aufweisen. Deswegen wurde für Ctrax ein eigenes Videoformat entwickelt, welches allerdings nicht an die Kompressionsraten anderer Formate heranreicht.

Die Nutzung dieses Videoformats ist im Biorobotics Lab aufgrund des erhöhten Speicherbedarfs nicht möglich. Für die Videos der Bienen ist ein Algorithmus, der auf Subtraktion des Hintergrundes basiert, ungeeignet, da sich der Hintergrund in Helligkeit und Farbgebung kaum von den Bienen unterscheidet. Ctrax sollte für die Videos der Hummeln und Fischen gute Ergebnisse liefern, aber die Geschwindigkeit des Trackings ist nicht ausreichend.

2.1.3 idtracker

Auch die Trackingsoftware idtracker basiert zu großen Teilen auf Matlab. Sie wurde zum Tracken von Fliegen, Mäusen und Fischen entwickelt, kann aber auch für alle anderen Tierarten benutzt werden, solange sich diese auf einem statischen Hintergrund bewegen.

Zur Vorverarbeitung wird beim idtracker keine 'Background Substraction' genutzt. Stattdessen werden alle Bilder zuerst in ihrer Intensität normalisiert, d.h. alle Pixelintensitäten werden durch den Durchschnittswert des Bildes geteilt. Dies soll Fehler durch Belichtungsfluktuation verhindern. Im nächsten Schritt werden 'Blobs' im Bild gesucht. Dies sind zusammenhängende Pixel, deren Lichtintensität über oder unter einer vorher vom Nutzer festgelegten Schwelle liegen. 'Blobs' die unter einer vom Nutzer festgelegten Mindestgröße liegen, werden verworfen. Die übrig geblieben 'Blobs' sollten nun den Objekten im Video entsprechen. Die Besonderheit an

dem Algorithmus hinter idtracker ist, dass für jedes Objekt im Video ein Fingerabdruck erstellt wird. Dieser besteht aus einer Reihe gefundener Bildausschnitte, die verschiedene Posen und Positionen des gleichen Objektes zeigen. Um Fingerabdrücke für alle Objekte zu finden, wird automatisch nach Stellen im Video gesucht, an denen alle Objekte, die im Video vorkommen, voneinander getrennt sind. Wird eine solche Stelle gefunden, können diese und alle nachfolgenden Bilder als Teile des Fingerabdrucks genutzt werden, bis die Anzahl an 'Blobs' nicht mehr der Anzahl der Objekte entspricht. Dies kann beispielsweise durch Kreuzen der Wege von Fischen in einem Tank passieren, da zwei Fische, die zu nah aneinander schwimmen, als ein 'Blob' erkannt werden könnten.

Durch die Technik der Fingerabdrücke können nun in allen Frames des Videos die Objekte gefunden und ihnen ihre korrekte einzigartige ID zugewiesen werden. Dieser Ansatz verhindert Fehlerpropagation, welche bei anderen Methoden auftritt.

Um eine gute Qualität an Fingerabdrücken zu erhalten, empfehlen die Entwickler eine Mindestgröße pro Objekt von 150px. Außerdem sollten Videos ungefähr eine FPS von 25 haben, damit sich die 'Blobs' eines Objektes in aufeinanderfolgenden Frames überlappen. Gute Ergebnisse konnten schon bei 5 Minuten langen Videos erreicht werden; empfohlen werden aber 30 Minuten, wenn die Zahl der Objekte maximal 35 ist, da zur Erkennung der 'Fingerabdrücke' Teile im Video benötigt werden, in denen alle Tiere erkannt werden können, ohne sich zu überlappen.

Bei der Einrichtung eines Experiments, welches idtracker nutzen soll, muss außerdem auf eine homogene Belichtung des Experiments geachtet werden. Enthalten Videos Stellen, die sich in der Belichtung merklich vom Rest des Videos unterscheiden, nimmt die Trackingquote hier ab. Dies liegt an der Normalisierung der Intensität und an der Art und Weise, wie Fingerabdrücke zu den Objekten zugeordnet werden.

Idtracker ist für die Videos der Bienen ungeeignet, da diese eine zu hohe Dichte an Tieren aufweisen und die Berechnung der Fingerabdrücke nur dann funktioniert, wenn Bilder im Video existieren, wo sich keine der Tiere überlappen. Die Videos der Hummeln weisen eine zu kleine FPS auf, wodurch sich die gefundenen 'Blobs' in aufeinanderfolgenden Frames nicht überlappen und deswegen die Berechnung der Fingerabdrücke nicht möglich ist. Gut geeignet ist der idtracker allerdings für die Videos der Fische, wobei der idtracker im Allgemeinen, um die guten Ergebnisse liefern zu können, ein hohes Maß an Berechnung durchführen muss und somit auch bei Videos mit wenigen Tieren kein Tracking in Echtzeit durchführen kann. Dies macht ihn für die Nutzung im Biorobotics Lab ungeeignet.

2.2 Interne Trackingprogramme

2.2.1 BeeDance Tracker

Der BeeDance Tracker ist ein von Landgraf *et al.* [6] entwickelter Tracker, der zur Verfolgung des für Honigbienen typischen Kommunikationsmittel 'Waggle Dance' genutzt wird. Zur Verfolgung des Objekts (Biene) wird vom Nutzer ein Bereich festgelegt, den das Objekt zu großen Teilen

ausfüllt. Für diesen Bereich, auch 'Bounding Box' genannt, werden Merkmale des optischen Flusses berechnet, welche im nächsten Frame wieder gesucht werden. Dabei wird angenommen, dass sich die Bewegung des Objekts durch eine rigide Transformation beschreiben lässt. Durch Erstellung von hypothetischen neuen Positionen der 'Bounding Box' wird die wahrscheinlichste neue Position berechnet. Hierfür wird die benötigte Transformation und Bewertung dieser hypothetischen Transformation mittels eines Hough Hash, siehe 3.2.1.2, berechnet. Eine genauere Beschreibung des Algorithmus ist unter 3.2.1 zu finden.

2.2.2 Fishtracker

Der Fishtracker ist ein Programm zum Tracken der Guppys im Robofish Projekt. Zuerst wurde ein Algorithmus implementiert, der mittels 'Background Substraction' und Blobdetektion die Fische im Video erkennt und mittels eines 'nearest neighbour'-Algorithmus die Pfade der Fische erstellt. Der 'nearest neighbour'-Algorithmus wurde nachträglich erweitert, indem der gefundene Blob mit einer kleinstmöglichen Ellipse umgeben wird und der Winkel der Ellipse in die Gewichtung der Nachbarn mit einfließt. Dies soll die Pfaderkennung verbessern, da die Guppys selten schnelle Drehungen durchführen. Der Algorithmus wird unter 3.2.2 genauer beschrieben. Zur Verbesserung der Ergebnisse wurden verschiedene Algorithmen basierend auf Partikelfiltern implementiert. Diese werden in dieser Arbeit nicht weiter behandelt.

Kapitel 3

3 Implementierung

In diesem Kaptiel werden zuerst die Anforderungen an den RigidFlow Tracker und den SimpleTracker behandelt. Danach werden die Implementierungen der Tracker besprochen, wobei ausführlich auf die Algorithmen eingegangen wird. Außerdem werden genutzte Fachbegriffe erläutert.

3.1 Anforderungen

Die beiden vorgestellten internen Trackingprogramme sollen in das BioTracker Framework portiert werden, um diese später als Beispiele für Tracker mit diesem zur Verfügung stellen zu können. Dafür müssen sie, wie das Framework, OpenSource gehalten werden.

3.1.1 RigidFlow Tracker

Die Implementierung des Algorithmus aus [6] liefert bereits qualitativ ausreichende Ergebnisse. Der Algorithmus soll in das BioTracker Framework portiert werden und ähnliche Ergebnisse liefern. Außerdem sollen Verbesserungen an den Eingabemöglichkeiten vorgenommen werden.

3.1.2 SimpleTracker

Aus dem Fishtracker soll der einfache Trackingalgorithmus, beruhend auf 'Background Substraction' und 'nearest neighbour'-Matching, entnommen und für die Nutzung im BioTracker angepasst werden. Hierfür müssen alle Parameter vom Nutzer einstellbar sein und die Bildverarbeitung auf die im BioTracker übliche Abfolge umgestellt werden.

3.2 Beschreibung der Trackingalgorithmen

3.2.1 RigidFlow Tracker

Um mit dem RigidFlow Tracker ein Objekt zu verfolgen, muss im ersten Frame ein kleinstmögliches Rechteck um das zu trackende Objekt gelegt werden. Dieser Bereich wird als 'Bounding Box' bezeichnet. Die Position, Größe und Orientierung des Objektes wird mit den Werten

der 'Bounding Box' gleichgesetzt. Nun werden die Merkmale des optischen Flusses der 'Bounding Box' berechnet. Diese werden im nächsten Frame gesucht und zur Berechnung der rigiden Transformation genutzt. Der Algorithmus nimmt an, dass die Bewegung eines Objekts durch eine Translation und Rotation der 'Bounding Box' beschrieben werden kann. Mit einem Hough Hash, siehe 3.2.1.2, wird diese rigide Transformation bestimmt. Durch Anwendung dieser wird die 'Bounding Box' im Frame verschoben und so die neue Position und Ausrichtung des Objektes festgelegt.

Im Tracker wird zwischen zwei Methoden des Trackings unterschieden, dem Automatischen und Semiautomatischen. Im Semiautomatischen werden die Merkmale des optischen Flusses nur im ersten Frame initialisiert. Dies führt wie in [6] beschrieben zur Verminderung der Qualität der Features nach einigen Frames und kann zu falschen Transformationen der 'Bounding Box' führen. Deswegen wird hier erwartet, dass der Nutzer bei zu schlechten Tracks die 'Bounding Box' manuell neu ausrichtet. Wenn die Box neu ausgerichtet wurde, werden die Merkmale neu berechnet und somit zur ursprünglichen Qualität zurückgesetzt. Nun kann wieder automatisch getrackt werden, bis wiederum ein qualitativ schlechter Track entsteht und die Box manuell verschoben werden muss.

Beim automatischen Tracking werden mehrere Folgen des optischen Flusses in die Berechnung der neuen Position einbezogen. Zusätzlich kann ein 'Autokorrektur-Schritt' aktiviert werden. Dieser soll durch eine aufwendigere Rechnung eine bessere Position der 'Bounding Box' finden. So kann der Nutzer das Tracking bedenkenlos autonom durchführen, denn der Autokorrekturschritt sollte eine zu starke Abnahme der Qualität der Tracks verhindern. Der Nutzer kann die Bounding Box trotz allem manuell verschieben, wenn dieses gewünscht ist.

3.2.1.1 Optischer Fluss

Optischer Fluss beschreibt die relative Bewegung eines Objekts in einer Szene zu dem Beobachter. Zum Verfolgen des optischen Flusses in den Videos werden mittels OpenCV's *goodFeaturesToTrack*-Funktion Merkmale im Bereich der Bounding Box gefunden. Im darauffolgenden Frame werden mit OpenCV's *calcOpticalFlowPyrLK*-Funktion die Merkmale im nächsten Frame wieder gefunden. Der optische Fluss beschreibt nun die Bewegung der Merkmale aus dem ersten Frame zum zweiten Frame. Diese Bewegung wird als rigide Transformation angenommen und mittels eines Hough Hash, siehe 3.2.1.2, bestimmt.

3.2.1.2 Hough Hash

Ein 'Hough Hash' ist ein von Landgraf *et al.* [6] definierter Begriff. Er beschreibt eine Funktion, die mittels einer Hough Transformation die rigide Transformation einer Bewegung bestimmt. Hierbei werden die Parameter der rigiden Transformation gehasht, um den Speicherbedarf zu reduzieren.

Um die rigide Transformation der Bounding Box zu bestimmen, werden im Hough Hash für jedes Merkmal des optischen Flusses die Parameter der Translation und Rotation berechnet. Dabei werden für jeden ganzzahligen Winkel zwischen -20° und 20° die Translationsparameter bestimmt. Der Winkel und die Parameter werden dann in einen Hash umgewandelt und ihre Bewertung wird um eins erhöht. Nachdem für alle Punkte die Transformationen berechnet und bewertet wurden, wird die Transformation angewendet, welche die höchste Bewertung hat. Haben mehrere Transformationen die selbe Bewertung, werden sie gemittelt und die Mittelung wird angewendet. Die daraus resultierende Bounding Box wird als neue Position des Objektes gesetzt.

Beim automatischen Tracking wird in jedem Frame ein neues Set an Merkmalen des optischen Flusses bestimmt. Vom Nutzer ist hierbei bestimmbar, nach wie vielen Frames ein Set verworfen wird. Dieser Wert bestimmt auch die Anzahl an Sets von Merkmalen, die gleichzeitig vorhanden sind. Der Hough Hash bewertet hier alle Punkte aller Sets mit der selben Methode.

3.2.1.3 automatische Korrektur der Bounding Box

Nach einer vom Nutzer festgelegten Anzahl an Frames, wird beim automatischen Tracking ein Autokorrekturschritt durchgeführt. Dieser soll die Verschiebung der Bounding Box vom Objekt ausgleichen, welche durch die abnehmende Qualität der Merkmale des optischen Flusses auftritt.

Hierfür werden 40 Modelle erstellt, welche jeweils eine Bounding Box mit zufälliger Position enthalten. Zur Bewertung jedes Modells wird wiederum ein Hough Hash genutzt. Hierbei wird für jedes Merkmal jedes Punktes erst einmal überprüft, ob es in der Bounding Box des Modells liegt. Dann wird wieder für jeden ganzzahligen Winkel zwischen -20° und 20° die rigide Transformation berechnet. Liegt das Merkmal in der Bounding Box, wird die Bewertung der Transformationen um eins erhöht. Liegt das Merkmal außerhalb, wird die Bewertung um eins verringert. Das Modell mit der besten Bewertung stellt nun die vorläufige neue Bounding Box des Elementes dar.

Dieser Schritt wird solange wiederholt, bis in den 40 erstellten Modellen keines eine bessere Bewertung hat, als das bestbewertete Modell aus dem letzten Schritt. Die Bounding Box dieses Modells wird als neue Position des Objektes genutzt.

3.2.1.4 Parameter

Der RigidFlow Tracker bietet dem Nutzer einige Einstellungsmöglichkeiten (Parameter) an, welche das Tracking oder die Ansicht der Tracks beeinflussen.

Mit den Radiobuttons 'Semi-automatic tracking' und 'Automatic Tracking' wird zwischen automatischen und semiautomatischen Tracking entschieden. 'Enable Correction' aktiviert den Korrekturschritt bei automatischem Tracking. Für den Korrekturschritt stehen zwei weitere Parameter zur Verfügung: 'Future Steps' und 'Non-Correction Steps'. 'Future Steps' legt die Anzahl von Frames fest, welche beim Korrekturschritt in die Berechnung einbezogen werden.

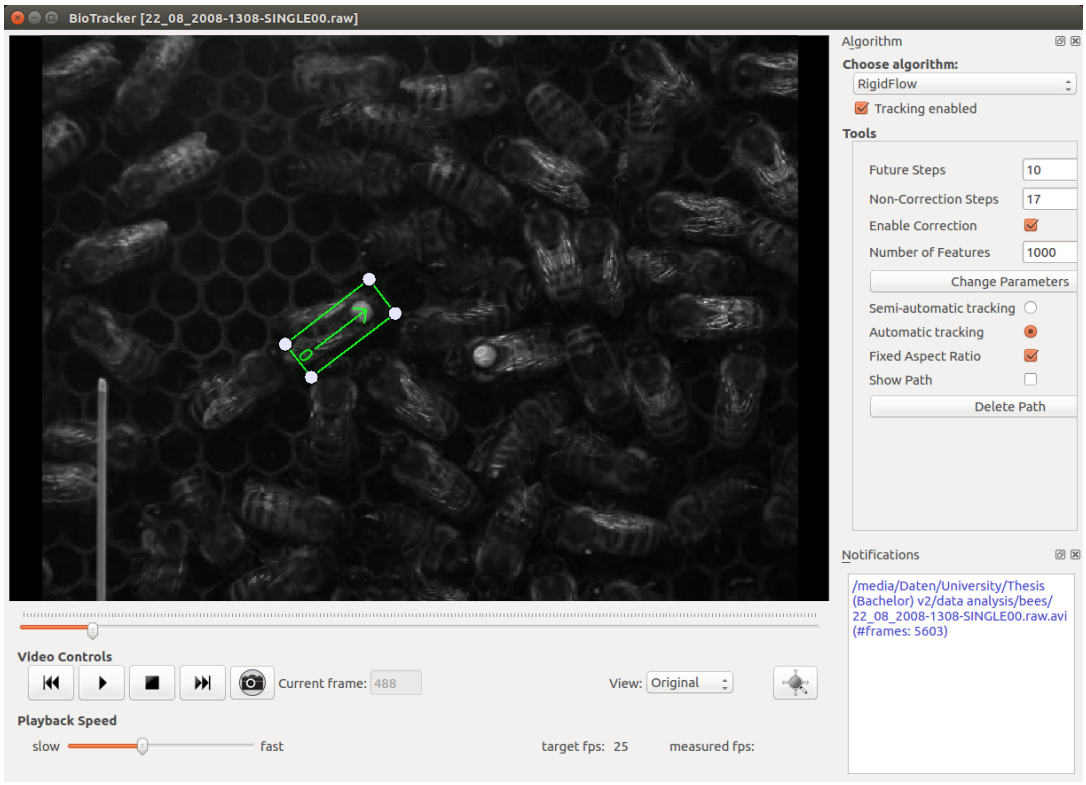

Abbildung 3.1: *Screenshot des BioTrackers mit geladenem Video und RigidFlow Tracker. Im Video ist die Biene, welche getrackt werden soll, mit einer grünen Box, der Bounding Box, markiert. Auf der rechten Seite können die Parameter des RigidFlow Trackers eingestellt werden.*

'Non-Correction Steps' gibt an, nach wie vielen Frames ein Korrekturschritt durchgeführt werden soll. Der Parameter 'Number of Features' gibt die Anzahl der Punkte des optischen Flusses an, welche beim Tracking berechnet werden sollen.

Mit dem Parameter 'Show Path' bietet der Tracker die Möglichkeit, vorherige und nachfolgende Tracks des momentan selektierten Objekts als Pfad anzuzeigen. Dieser wird als hellgrüne Linie für nachfolgende und dunkelgrüne Linie für vorherige Tracks angezeigt. 'Fixed Aspect Ratio' ermöglicht es dem Nutzer, das Verhältnis zwischen Höhe und Breite der Bounding Box festzulegen. Dies hilft beim Festlegen der Bounding Boxen mehrerer Objekte, da bspw. die Bienen wie in 3.1 zu sehen, meist das selbe Verhältniss von Länge zu Breite aufweisen.

3.2.2 SimpleTracker

Der 'SimpleTracker' ist ein 'einfacher' Tracker, welcher vollautomatisch alle sich bewegenden Objekte in einem Video erkennen und ihren Pfad verfolgen soll. Zur Erkennung der Objekte wird zuerst der Hintergrund entfernt und dann die verbleibenden zusammenhängenden Konturen

erkannt. Diese werden an die Pfade bereits vorhandener Objekte angehängt, oder ein neues Objekt wird angelegt. Zur Anwendung der genutzten Algorithmen wird jedes Bild zuerst in ein Schwarz-Weiß-Bild umgewandelt.

3.2.2.1 Background Subtraction

Der Tracker erwartet kein Bild, welches ausschließlich den Hintergrund enthält. Daher muss zur Entfernung des Hintergrunds zuerst der Hintergrund berechnet werden. Hierfür wird im ersten Frame das Bild als Hintergrund angenommen. In jedem darauffolgenden Frame wird nun der momentane Hintergrund gewichtet mit dem aktuellen Frame addiert. Unter der Annahme, dass sich die Objekte ständig im Verlaufe des Videos bewegen, führt dies zur Entfernung der Objekte aus dem Bild und somit zur Erstellung einer guten Näherung eines echten Hintergrundbildes. Bleiben Objekte jedoch über einige aufeinanderfolgende Frames an der selben Stelle stehen, können diese in den Hintergrund übernommen werden. Dadurch werden sie nicht mehr als Objekte erkannt.

3.2.2.2 Filtern und Erkennung

Nach der 'Background Substraction' entsteht ein Bild des Vordergrundes, welches an Stellen des Hintergrundes den Wert 0 haben sollte. Durch verschiedene Störfaktoren, wie die Veränderung der Lichtverhältnisse im Verlaufe eines Videos, Rauschen oder die nicht optimale Approximation des Hintergrundes, kann es Stellen im Vordergrundbild geben, welche eigentlich Teil des Hintergrundes sind, aber als Vordergrund erkannt werden. Um diese Stellen zu filtern, wird ein Schwellwert festgelegt. Alle Werte, die kleiner als dieser Schwellwert sind, werden auf 0 gesetzt, und somit als Teil des Hintergrundes angenommen. Um eventuelles Rauschen zu entfernen, wird eine bestimmte Anzahl an Erosionen durchgeführt, wodurch es zur Trennung von Konturen kommen kann. Zur Rekombination dieser, kann eine Anzahl an Dilatationen durchgeführt werden.

Um nun die Objekte zu finden, wird OpenCv's *findContours* Methode genutzt. Diese gibt eine Menge an gefunden Konturen zurück. Zur weiteren Verbesserung der Erkennung, kann eine minimale und maximale Größe der Konturen vorgeschrieben werden. Nach diesen Vorgaben werden die Konturen nun gefiltert. Für die übriggebliebenen Konturen werden kleinstmögliche Ellipsen gefunden, welche die Objekte repräsentieren sollen.

3.2.2.3 Pfadzuweisung

Zur Zuweisung einer gefunden Kontur zum richtigen Pfad wird ein modifizierter 'nearest neighbour'-Algorithmus genutzt. Im ersten Schritt werden alle Pfade für eine mögliche Zuweisung entfernt, die keinen Track im letzten Frame hatten. Für jeden der übrigen Pfade wird nun die Kontur gesucht, die am Besten passt. Dabei wird, aufgrund der Daten des letzten Frames, die Position des Objektes in diesem Frame abgeschätzt. Diese Schätzung wird dann mit allen Konturen verglichen, wobei die Position und der Winkel der Schätzung und der Kontur in den Vergleich einfließen. Für die Kontur, die am Besten zum Objekt passt, wird nun überprüft, ob

diese zu einem anderen Objekt besser passt. Ist dies nicht der Fall, wird die Kontur als Position dieses Frames dem Objekt zugeordnet. Sollte die Kontur besser zu einem anderen Objekt passen, wird die Kontur vorerst keinem Objekt zugeordnet. Die Suche nach der besten Kontur wird mit dem Objekt fortgesetzt, zu welchem die Kontur besser passt. Ist die Kontur die Beste für dieses Objekt, würde aber besser zum Objekt aus dem vorherigen Schritt passen, wird die Kontur trotzdem dem momentanen Objekt zugeordnet, um eine endlose Schleife zu verhindern. Diese Schritte werden solange wiederholt, bis alle Konturen zu einem Objekt hinzugefügt wurden oder alle Objekte eine Kontur zugeordnet bekommen haben.

Sind nach der Zuordnung noch Konturen übrig, werden diese zu Kandidaten umgewandelt. Diese Kandidaten sind eine Vorstufe zu einem gefunden Objekt. Kandidaten werden nach einer gewissen Anzahl von aufeinanderfolgenden Detektionen zu Objekten befördert.

3.2.2.4 Parameter

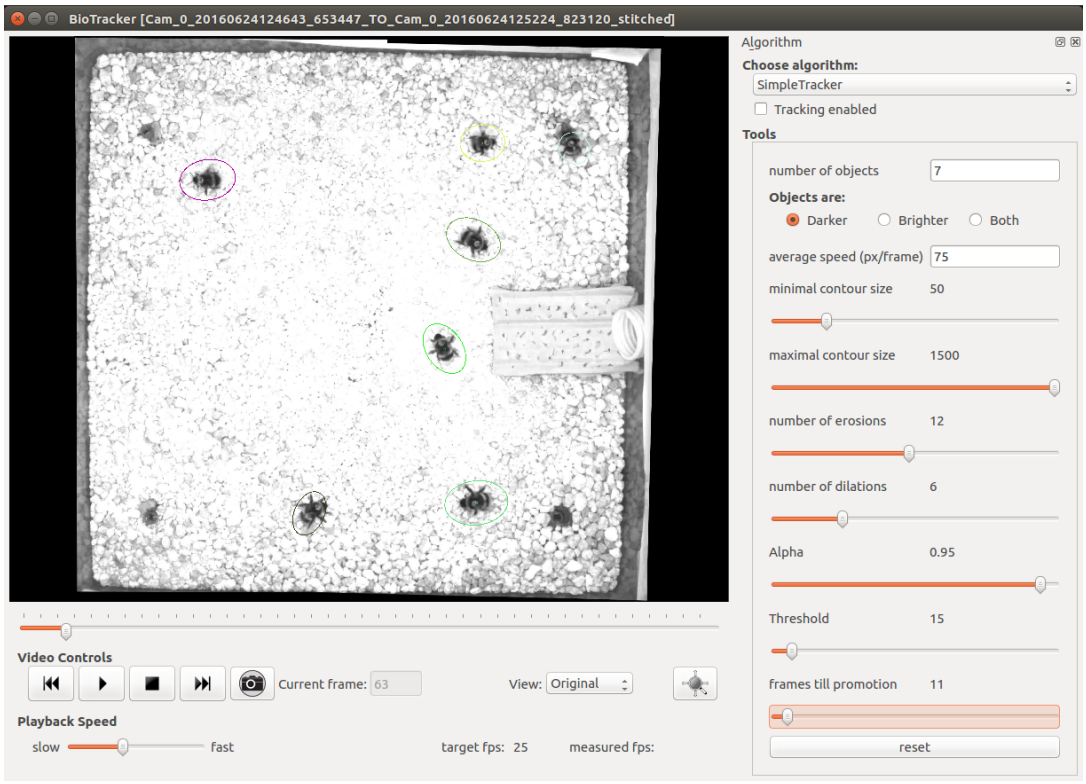

Abbildung 3.2: *Screenshot des BioTrackers mit geladenem Video und SimpleTracker. Im Video sind Hummeln auf Futtersuche zu sehen. Die farbigen Ellipsen stellen die gefundenen Hummeln dar. Auf der rechten Seite können die Parameter des SimpleTrackers eingestellt werden.*

Damit der SimpleTracker auf verschiedenste Videos angewendet werden kann, stellt dieser verschiedenste Einstellungen (Parameter) zur Verfügung. Der Parameter 'number of objects' gibt die maximale Anzahl von Objekten, die in einem Frame zu tracken sind, an. Der Nutzer sollte festlegen, ob die Objekt dunkler oder heller als der Hintergrund sind. Diese Auswahl hat Einfluss auf die Art und Weise, wie der Hintergrund abgezogen wird. Es besteht auch die Möglichkeit, das Objekte heller und dunkler als der Hintergrund sind. 'average speed' gibt die durchschnittliche Entfernung an, die ein Objekte in einem Frame zurücklegt. Dieser Parameter wird für die Zuweisung der Tracks zu den Pfaden benötigt.

Um Tracks, welche viel größer oder viel kleiner als die erwartete Größe eines Objekts sind, auszuschließen, sind die Parameter 'minimal contour size' und 'maximal contour size' vorhanden. 'number of erosions' und 'number of dilations' geben die Anzahl der Erosionen und Dilationen and. 'Alpha' legt den Einfluß des momentanen Hintergrundbildes auf das nächste Hintergrundbild an. 'Threshold' bestimmt die Helligkeiten, die nach der Subtraktion des Hintergrundbildes verworfen werden. Die Anzahl der benötigten aufeinanderfolgenden Tracks, die zur Beförderung eines Kandidaten zum Objekt führen, wird durch 'frames till promotion' festgelegt.

Kapitel 4

4 Evaluierung

In diesem Kapitel werden die Benutzbarkeit und Zuverlässigkeit der Tracker analysiert. Bei der Analyse der Zuverlässigkeit wird auf typische Kriterien von Trackingalgorithmen, wie Genauigkeit der gefundenen Position, falsche und fehlende Detektionen, sowie Fragmentierung, eingegangen.

4.1 Benutzbarkeit

Nachdem die Tracker fertig implementiert waren, wurden diese auf ihre Benutzbarkeit und Robustheit geprüft. Beim RigidFlow Tracker hat sich das Anpassen der Ausrichtung als ein Streitpunkt herauskristallisiert. Durch Betätigen der rechten Maustaste kann die Ausrichtung angepasst werden. Hierfür wird der Winkel zwischen dem Start- und dem Endpunkt der Bewegung in Relation zum Mittelpunkt der Bounding Box berechnet und die Box um diesen gedreht. Einige Nutzer würden allerdings eine eindimensionale Variante bevorzugen, in der nur der Abstand in x- oder y-Richtung zwischen Start- und Endpunkt zum Bestimmen des Winkels genutzt wird.

Beim SimpleTracker wirft die Einstellung der Parameter Probleme auf. Durch die schiere Anzahl und die Tatsache, dass ohne Hintergrundwissen über den Algorithmus die Auswirkungen der Anpassung eines Parameters nicht abzusehen sind, stellte sich das Finden von Parametern, die qualitative gute Ergebnisse liefern, als schwierig heraus.

4.2 Zuverlässigkeit

Für die Bestimmung der Zuverlässigkeit der Trackingalgorithmen werden beispielhaft zwei Videos, eines für jeden Algorithmus, analysiert. Hierfür werden zuerst Vergleichsdaten erstellt. Mithilfe des semiautomatischen Trackings des RigidFlow Trackers werden Größe, Position und Ausrichtung der Objekte festgelegt und in Pfade zusammengeschlossen, sodass für jedes Objekt ein Pfad existiert, welcher die Bewegung vom Eintritt in den Experimentierbereich bis zum Austritt beschreibt. Hierbei wird die Box sofort verschoben, wenn sie sich minimal von der Biene oder Hummel bewegt.

Beide Trackingalgorithmen erwarten vom Nutzer vorgegebene Parameter. Diese können die Ergebnisse des Trackings stark beeinflussen und sind in 3.2.1.4 und 3.2.2.4 beschrieben. Deswegen werden die Videos mehrmals, mit verschiedenen Parametern, durchlaufen, bis keine relevanten Verbesserungen der Ergebnisse festgestellt werden können.

4.2.1 RigidFlow Tracker

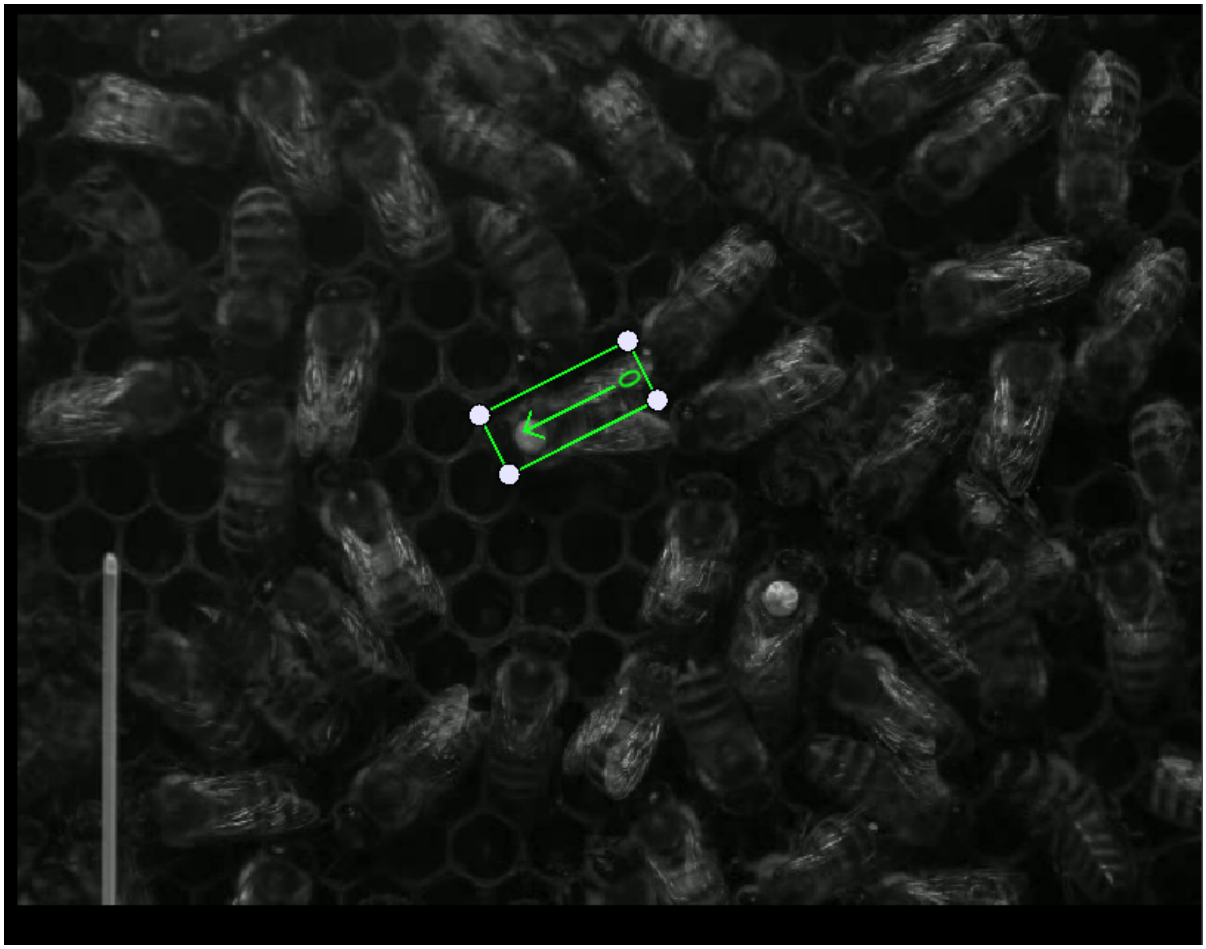

Abbildung 4.1: *Ein Bild des Videos, welches mit dem RigidFlow Tracker getrackt wurde. In dem Video bewegen sich Bienen auf einer Wabe. Die Biene, welche mit der grünen Bounding Box markiert ist, führt gerade einen Tanz aus.*

Bei dem Analysieren eines Videos mit dem RigidFlow Tracker muss für jedes Objekt, das getrackt werden soll, eine Bounding Box erstellt werden. Da der Algorithmus für jedes Bild im Video eine neue Position der Bounding Box festlegt, können keine Detektionen fehlen oder eine Fragmentierung des Pfades auftreten, sodass die Qualität des Trackings allein durch die Position und die Ausrichtung beschrieben wird.

Um die genutzten Parameter der gezeigten Ergebnisse zu symboliseren, wird die Notation 'Future Steps'/'Non-Correction Steps'/'Number of Features', bspw. 10/10/5000, genutzt. Alle Durchläufe wurden mit automatischem Tracking und aktiviertem Korrekturschritt durchgeführt.

4.2.1.1 Räumliche Fehler

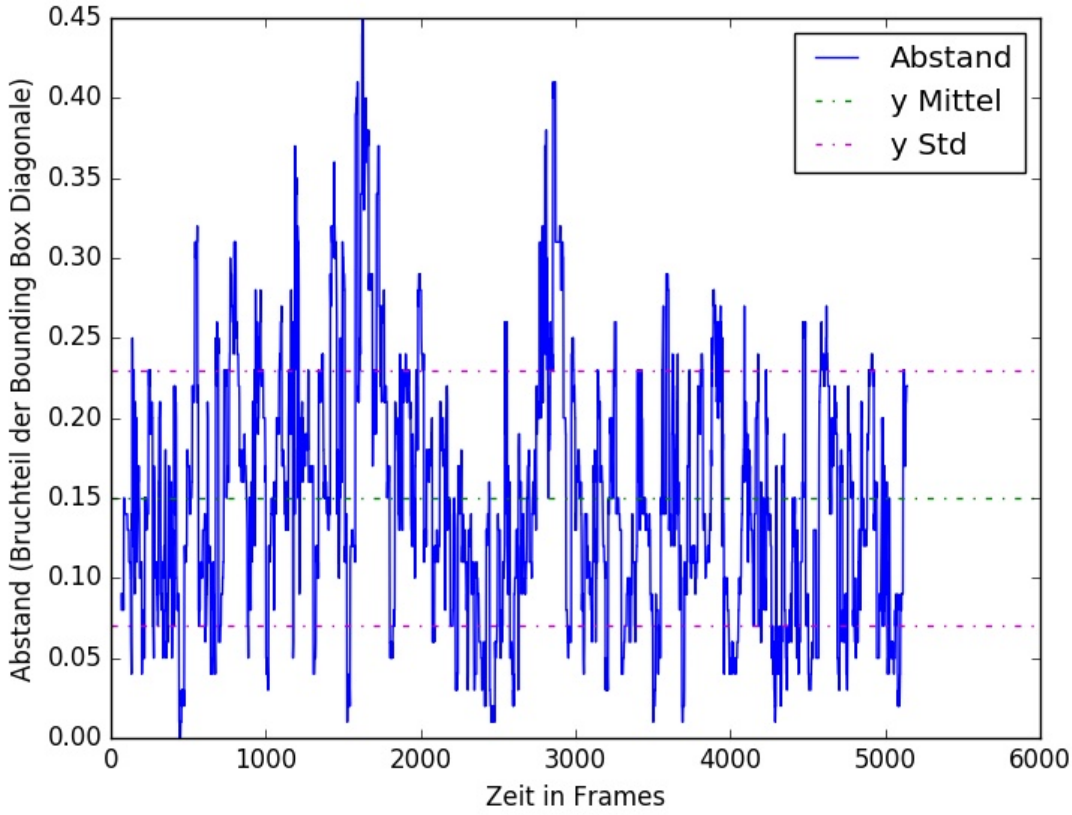

Abbildung 4.2: *Relative Abstände der Positionen vom automatischem zu manuellem Tracking für die Konfiguration 10/17/1000. Mittelwert: 0,15, Standardabweichung: 0,08.*

Die Berechnung des Positionsfehlers wird über den euklidischen Abstand durchgeführt. Da in anderen Videos die Bienen eine größere oder kleinere Pixelfläche einnehmen können, je nachdem wieweit die Kamera von der Wabe entfernt ist und in welcher Auflösung die Kamera aufnimmt, wird die Differenz der Position in Relation zur Diagonale der Bounding Box gestellt. Die kleinste durchschnittliche Differenz konnte mit den Parametern 10/10/5000 festgestellt werden und liegt bei 0,14 mit einer Standardabweichung von 0,07. Die Differenzen der Konfigurationen

10/15/1000 und 10/17/1000 liegen nur wenig darüber bei 0,15; mit einer Standardabweichung von 0,08. Alle Werte sind auf zwei Nachkommastellen gerundet. Andere getestete Konfigurationen liefern weitaus höhere Werte. Sie sind in A.1 aufgelistet.

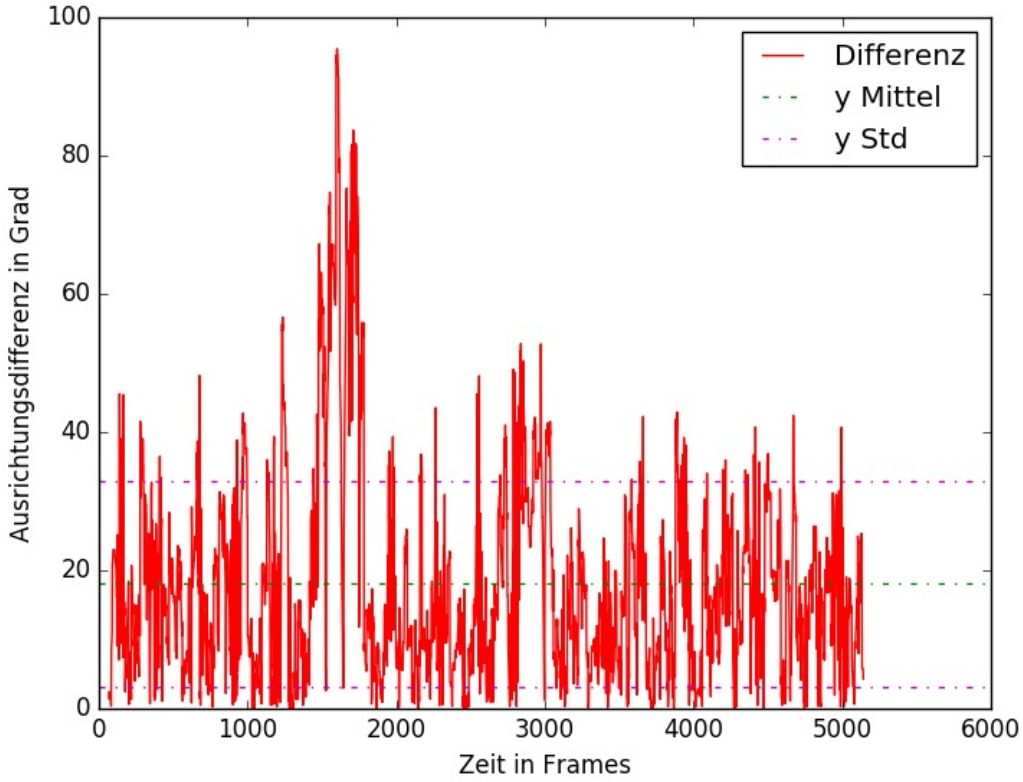

Abbildung 4.3: *Absolute Differenzen der Ausrichtungen vom automatischem zu manuellem Tracking für die Konfiguration 10/17/1000. Mittelwert: 18,06, Standardabweichung: 14,94.*

Die Qualität der Ausrichtung wird durch die Berechnung der absoluten Differenz der Ausrichtung in den manuell getrackten und den automatisch getrackten Daten gemessen. Der kleinste durchschnittliche Wert wurde mit der Konfiguration 10/17/1000 berechnet und beträgt 18,06 mit einer Standardabweichung von 14,94. Alle Werte sind auf zwei Nachkommastellen gerundet. Andere getestete Konfigurationen liefern weitaus höhere Werte. Sie sind in A.2 aufgelistet.

Der hier berechnete Wert für den Positionsabstand 0,15 unterscheidet sich nur leicht von den Werten, die von Landgraf *et al.* [6] berechnet wurden (0,05-0,13); die Ausrichtungsdifferenz hier unterscheidet sich allerdings mit 18° stark von 7°-12°.

4.2.2 SimpleTracker

Der SimpleTracker benötigt als Eingabe nur ein Video und die Parameter, die zum Tracking genutzt werden sollen. Beim Tracking werden Unterschiede zwischen dem Bild und dem Hintergrund gesucht, die möglichen Tracks. Diese werden bereits vorhandenen Pfaden zugeordnet, oder es werden ggf. neue Kandidaten erstellt.

Diese Vorgehensweise ermöglicht verschiedenste Fehlerquellen. Objekte können nicht gefunden werden; Stellen im Bild können sich vom Hintergrund unterscheiden, obwohl sich dort kein Objekt befindet. Dies führt zu falschen Detektionen. Die Pfadzuweisung kann fehlerhaft sein, was zu Pfaden, welche mehrere Objekte enhtalten, oder zu Fragmentierung, also der Zerlegung eines Pfades in Mehrere, führen kann.

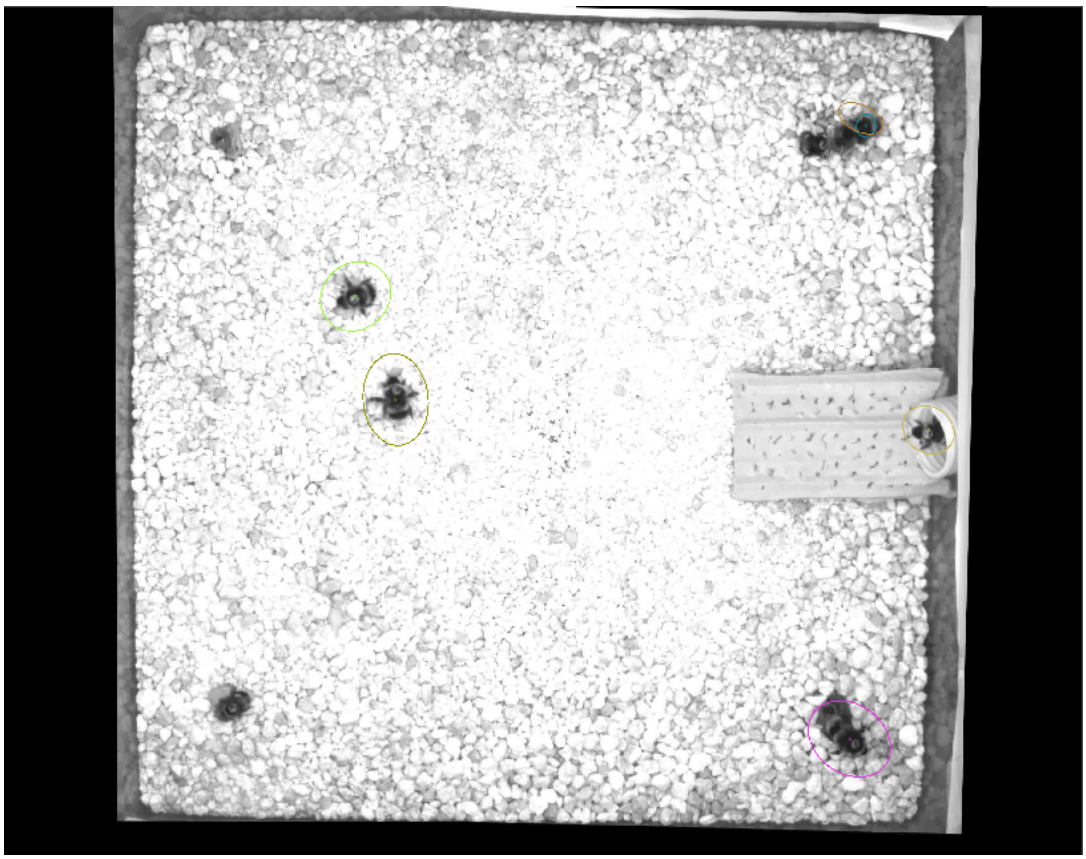

Abbildung 4.4: *Ein Bild des Videos, welches mit dem SimpleTracker getrackt wurde. In dem Video bewegen sich Hummeln in einer Box mit vier möglichen Futterstellen. Die farbigen Ellipsen stellen gefundene Objekte dar. Eine falsche Detektion ist im rechten oberen Teil des Videos zu beobachten, hier überlappen sich zwei Ellipsen.*

Alle Konfigurationen werden mit 'maximal contour size' von 1500, einem 'alpha' von 0,95, sowie einem 'Threshold' von 15 und 'frames till promotion' von 10 ausgeführt. Die Objekte sind dunkler als der Hintergrund. Die restlichen Parameter werden im Folgenden als 'number of object'/'average speed'/'minimal contour size'/'number of erosions'/'number of dilations' dargestellt.

Für genauere Analysen wird die Konfiguration 7/75/25/12/6 (Nr. 2 in A.3) betrachtet, da mit dieser über alle Kriterien die besten Ergebnisse erreicht wurden.

4.2.2.1 Fehlende und falsche Detektionen

Als fehlende Detektionen werden jene bezeichnet, die in den manuell erstellten Daten vorhanden sind, in den Automatischen allerdings nicht gefunden wurden. Das beste Ergebnis konnte mit der Konfiguration 7 (8/75/25/8/5) erzielt werden. Es wurden 4989 der manuell erstellten Detektionen gefunden, welches 89% entspricht. Die Konfigurationen 1, 2, 3, 6, 8, 9, 12 und 13 liefern ebenfalls Ergebnisse über 80%.

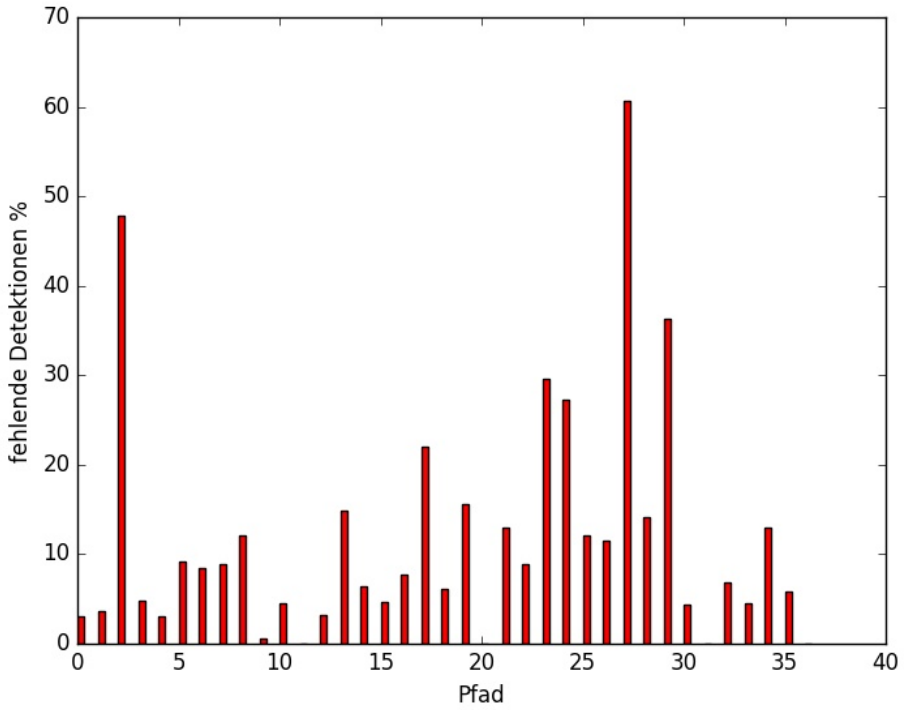

Abbildung 4.5: *Im Diagramm sind die fehlenden Detektionen je Pfad dargestellt, welche in Prozent zu der Länge des Pfades angegeben werden. Auffällig sind hierbei die Ergebnisse der Pfade 2, 17, 23, 24, 27 und 29, da die Fehlerquote dort weit über denen der Anderen liegt.*

Falsche Detektionen sind jene, die in den automatisch erstellten Daten gefunden wurden, aber nicht in den manuell erstellten Daten zu finden sind. Die 3. Konfiguration enthält 2005 und damit 30%, solcher Detektionen. Ergebnisse zwischen 30% und 40% werden außerdem in den Konfigurationen 1, 2, 8 und 9 beobachtet.

In 4.5 sind die fehlenden Detektionen beim Tracking mit 7/75/25/12/6 nach Pfaden aufgeteilt. Bei einem Großteil der Pfade wird eine Fehlerquote von unter 20% festgestellt. Einige Pfade hingegen weißen Fehlerquoten von bis zu 60% auf.

4.2.2.2 Räumliche Fehler

Zur Einschätzung des räumlichen Fehlers wird der euklidische Abstand zwischen manuellen und automatischen Tracks, sowie die Differenz der Ausrichtung dieser, genutzt. Der euklidische Abstand wird außerdem in Relation zur Diagonalen des Rechteckes gesetzt, welche das Objekt bei den manuell erstellten Daten umschließt.

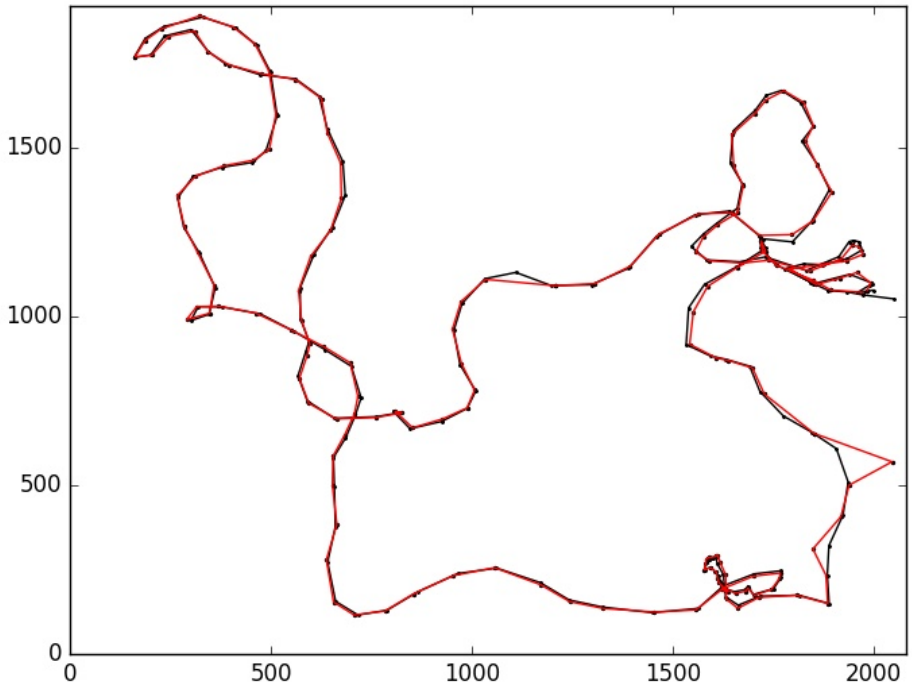

Abbildung 4.6: *Automatisch (rot) und manuell (schwarz) getrackter Pfad einer Hummel im Video. Der automatische Pfad besteht aus mehreren Fragmenten, die vom Tracker erkannt wurden. Einheiten der Achsen sind Pixel.*

Der durchschnittliche relative euklidische Abstand liegt bei allen Konfigurationen zwischen 0,1 und 0,12. Die Konfigurationen 1, 2, 3, 7, 8 und 9 liefern einen relativen Abstand kleiner gleich 0,11 und damit die niedrigsten Ergebnisse. Bei der Differenz der Ausrichtung erzielen allerdings die Konfiguration 4, 5, 12 und 13 die besten Ergebnisse mit einer Differenz von unter 68. Die anderen Konfigurationen erreichen Werte von bis zu 74,02. Die Liste aller Ergebnisse ist unter A.5 zu finden.

4.2.2.3 Fragmentierung

Um die Fragmentierung der automatisch erstellten Daten einschätzen zu können, werden die gefundenen Detektionen mit denen aus den manuell erstellten Daten verglichen und so zu Pfaden zusammengeführt. Diese Pfade enthalten dann alle richtigen Detektionen und können auf ihre Vollständigkeit überprüft werden.

In den manuell erstellten Daten existieren für das Video 36 Pfade mit insgesamt 5598 einzelnen Detektionen. Die Anzahl der Fragmente in den zusammengeführten automatischen Daten liegt zwischen 491 (Konfiguration 3) und 1170 (Konfiguration 12). Werden diese nun in Relation zu den richtigen Detektionen gestellt, lässt sich die durchschnittliche Länge eines Fragmentes bestimmen. Diese Werte liegen zwischen 9,02 (Konfiguration 9) und 3,93 (Konfiguration 12). Werte für alle Konfigurationen sind in A.6 aufgelistet.

Kapitel 5

5 Diskussion

Das Kapitel 'Diskussion' dient der Einordnung der Ergebnisse aus Kapitel 4. Hier werden die Qualität der Ergebnisse und ihre Verwendbarkeit in der Praxis diskutiert.

5.1 RigidFlow Tracker

Bei der Benutzung des RigidFlow Trackers treten kaum Probleme auf. Die Problematik der Ausrichtung der Box, die in 4.1 dargelegt ist, kann durch die Implementierung beider Varianten gelöst werden. Es muss dann eine Einstellung zur Verfügung stehen, die die Festlegung auf eine der Varianten übernimmt.

Die Qualität der Ergebnisse des Trackings ähneln denen, die Landgraf *et al.* [6] festgestellt haben, wobei sich der Fehler der Ausrichtung doch stärker unterscheidet. Dies könnte an dem gewählten Video liegen. In 4.2 und 4.6 kann eine gravierende Verschlechterung der Ergebnisse zwischen ungefähr Frame 1500 und 1800 festgestellt werden. Im Video beginnt die Biene im Frame 1450 zu tanzen, hierbei verschiebt sich die Bounding Box auf die rechte Seite der Biene und wärend dieser Tanzbewegung und der Nächsten wird die Box vom Algorithmus auf der Biene verschoben, wobei der Mittelpunkt der Box sich zwar durchgehend auf der Biene befindet, dieser aber durch Korrekturschritte auf der Biene hin und her bewegt wird. An dieser Stelle schlägt das Tracking offensichtlich fehl. In Frame 1801, nach einem Korrekturschritt, wird die Position der Biene wieder gefunden. Wird diese Stelle in den Ergebnissen ignoriert, verbessert sich der durchschnittliche Abstand leicht von 0,15 zu 0,14 mit einer Standardabweichung von 0,07, vorher 0,08. Die durchschnittliche Differenz der Ausrichtung verbessert sich merklich von 18,06° zu 15,53° während die Standardabweichung sich von 14,94° zu 10,71° ändert. Diese Werte liegen näher an denen von Landgraf *et al.* [6]. Der restliche Unterschied könnte an Veränderungen am Experiment (Kameras, Entfernung zur Wabe, etc.) liegen.

Im Allgemeinen wurde die Portierung des RigidFlow Trackers erfolgreich umgesetzt. Einige Verbesserungen bei der Benutzbarkeit und des Algorithmus sind aber weiterhin möglich.

5.2 SimpleTracker

Das größte Problem bei der Benutzung des SimpleTrackers ist das Finden der richtigen Parmater. Eine Lösung dieses Problems wird in 5.3 diskutiert.

Die Qualität der Ergebnisse des SimpleTrackers ist sehr durchwachsen. Bei guten Parameterkonfiguratioenen können bis zu 90% der Objekte gefunden werden, allerdings werden dabei auch bis zu 40% falsche Detektionen produziert. Damit sind die Detektionen des SimpleTrackers nicht verlässlich. Diese Ergebnisse könnten vermutlich durch eine Verbesserung der Erstellung des Hintergrundbildes verbessert werden. Momentan wird beim Start des Trackings das Bild des ersten Frames als Hintergrundbild benutzt, wodurch keinerlei Detektionen im ersten Frame auftreten. Im weiteren Verlauf werden alle getrackten Bilder bis einschließlich dem aktuellen Frame in das Hintergrundbild einbezogen. Dies geschieht durch gewichtete Addition der Bilder. Ein Objekt, welches sich längere Zeit auf einer Stelle befindet, geht dadurch in den Hintergrund über und kann nicht mehr gefunden werden. Die Bewegung eines Objektes sorgt außerdem für einen 'Schweif' im Hintergrundbild, d.h. es bildet sich ein Schatten der Bewegung im Hintergrundbild. An diesen 'Schweifen' kommt es immer wieder zu falschen Detektionen, da diese einen merklichen Unterschied zum Hintergrund aufweisen. Würde das Hintergrundbild vom Nutzer zur Verfügung gestellt, würden die fehlenden Detektionen, aufgrund fehlender Bewegung, nicht mehr auftreten und die Fehldetektionen würden vermutlich reduziert. Kann kein Hintergrundbild zur Verfügung gestellt werden, könnte das Hintergrundbild vor dem Start des Trackings berechnet werden. Ctrax beispielsweise wählt zufällig einige Bilder des Videos und berechnet aus ihnen den Hintergrund.

Richtige Detektionen sind im Durschnitt 10% der Länge der Diagonale des umschließenden Rechtecks vom manuell erstellten Punkt entfernt. Dies ist ein ausreichendes Ergebnis. In 5.1 ist außerdem abzulesen, dass sich ein Großteil der Abstände im Bereich unter 0,1 befinden, ein großer Teil dieser liegt sogar unter 0,05. Der Fehler der Ausrichtung im behandelten Video ist wiederum sehr hoch. Ein durchschnittlicher Fehler von 68° ist nicht zufriedenstellend. Allerdings liegt dieses mit großer Wahrscheinlichkeit am gewählten Experiment. Die Hummeln, welche sich im Video bewegen, weisen fast eine kreisförmige Kontur auf. Durch die Erosionen und Dilationen kann sich leicht die Ausrichtung der Kontur verändern, wodurch sich die schlechteren Ergebnisse erklären lassen. Zur Bestätigung der Annahme sollten weitere Videos analysisert werden, welche Objekte beinhalten, die ein größeres Verhältnis von Höhe zur Breite der Kontur aufweisen.

Die Fragmentierung in den Ergebnissen des SimpleTrackers ist ein weiteres Problem. Nachdem die Ergebnisse des Trackers zu Pfaden zusammengesetzt wurden, die denen der manuellen Daten entsprechen, weißen diese 511 Fragmente auf. In Relation zu den 36 Pfaden der manuellen Daten bedeutet dies, dass ein Pfad durchschnittlich in 14,19 Fragmente zerlegt ist. Ein solches Fragment hat durchschnittlich eine Länge von 9,08. Dies ist viel zu klein und ermöglicht kaum eine sinnvolle Analyse der Daten.

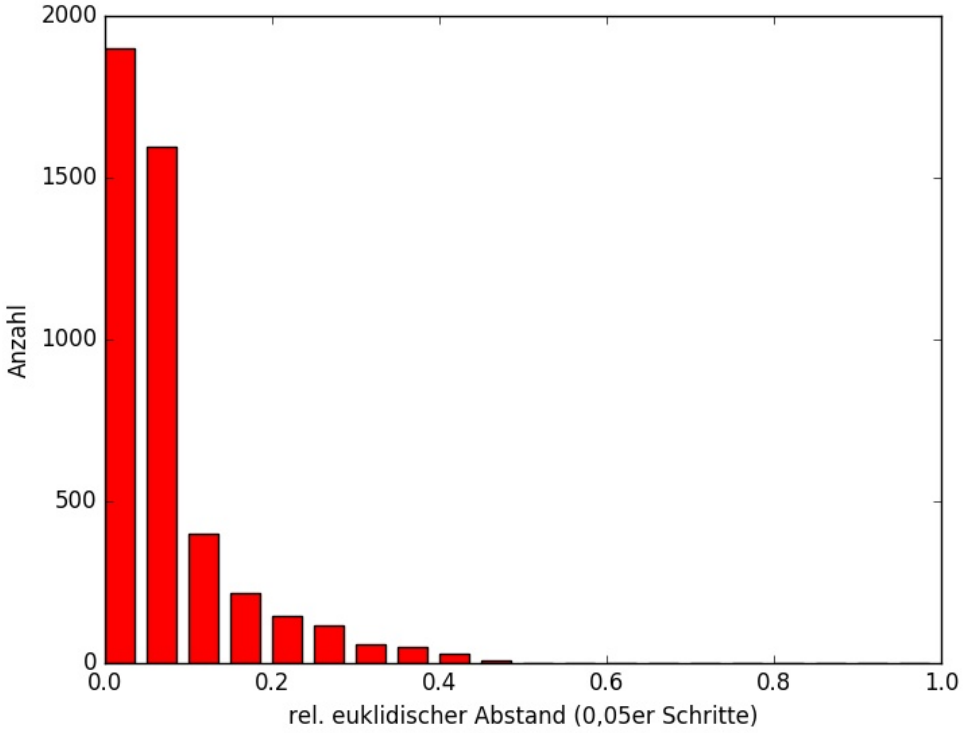

Abbildung 5.1: *Die festgestellten relativen Abstände beim Tracking mit dem SimpleTracker mit den Paramtern 7/75/25/12/6. Die Abstände wurden in 0,05er Schritte eingeteilt. Zu beobachten ist der exponentielle Abfall der Anzahl bei steigendem Abstand.*

5.3 Parameterauswahl

Um mit dem Tracking der Videos zu beginnen, ist es in beiden Trackern notwendig, Parameter festzulegen. Dies stellt sich als eine schwierige Aufgabe heraus, da der Einfluss der Parameter schwer abzusehen ist. Um die Auswahl der Parameter zu vereinfachen, kann man sich eine automatisierte Hilfe vorstellen. Beispielsweise könnte bei dieser ein Wertebereich der Parameter angegeben werden, sowie ein Standardwert und eine Sprungweite. Der BioTracker könnte dann mit verschiedenen Parameterkombinationen einen Teil des Videos analysieren, und die Ergebnisse dem Nutzer zeigen. Dieser könnte dann jene Kombination auswählen, die visuell die besten Ergebnisse liefert. Alternativ könnte der Nutzer auch einen Teil des Videos manuell tracken, sodass im BioTracker selbst entschieden werden kann, welche Konfiguration die Beste ist. Erweitern liese sich dies durch die Einbeziehung mehrerer Videos, da für ein Experiment, welches bearbeitet wird, meist einige Videos analysiert werden müssen. Dies könnte eine Konfiguration liefern, die für alle Videos brauchbar ist.

Kapitel 6

6 Ausblick

6.1 BioTracker Framework

Die Entwicklung am BioTracker Framework lief während der Portierung der Algorithmen weiter und es wurde eine neue Version entwickelt. In ihr wurden Probleme mit der Kommunikation innerhalb des Programmes, sowie einige Probleme mit der Struktur des Programms gelöst. Mit dieser neuen Version hat sich Einiges an der Struktur eines Trackers verändert. Die beiden hier vorgestellten Tracker werden in naher Zukunft in die neue Version portiert und eventuelle Verbesserungen werden sich vorrangig auf die neue Version des BioTrackers beschränken.

Während der Entwicklung des SimpleTrackers wurde die Möglichkeit diskutiert, das Video innerhalb des Trackers zu homogenisieren, d.h. mit Hilfe von Eingaben des Benutzers bspw. die Perspektive herauszurechnen. Es wurde hier allerdings beschlossen, dass dies eine Funktionalität sei, die für viele Tracker sinnvoll sein könnte, weswegen diese direkt in das BioTracker Framework integriert werden soll.

6.2 RigidFlow Tracker

Schon vor der Portierung des RigidFlow Trackers wurden Verbesserungen für das Tracking in diesem diskutiert. Eine davon ist die Anpassung des Bewegungsmodells, denn die Bienen, für den der Tracker entwickelt wurde, bewegen sich nicht immer rigide über die Wabe. Drehungen oder Krümmungen ihres Körpers sind keine Seltenheit und werden im momentanen Modell nicht behandelt. Diese Bewegungen mit einzubeziehen könnte weitaus bessere Ergebnisse liefern. Ob diese Anpassung in diesem Tracker durchgeführt wird, oder ein Neuer entwickelt wird, wurde noch nicht ausdiskutiert.

6.3 SimpleTracker

Das Tracking des SimpleTrackers ist im Allgemeinen noch zu ungenau. Die in 5.2 diskutierten Verbesserungen sollten durchgeführt werden, damit der Tracker besser nutzbar ist. Außerdem wird eine Dokumentation der Parameter benötigt, sodass auch Nutzer, die sich nicht mit dem Algorithmus beschäftigt haben, diese sinnvoll einstellen können.

6.4 Veröffentlichung BioTracker

Nachdem die beiden hier vorgestellten Tracker in die neue Version des Biotrackers überführt wurden und die neue Version des BioTrackers vollständig implementiert wurde, soll der BioTracker veröffentlicht werden. Momentan stehen zwar schon alle Implementierungen in öffentlichen github-Repositories zur Verfügung, allerdings müssen diese selbst kompiliert werden, welches die Installation aller Abhängigkeiten erfordert und somit einen großen Aufwand mit sich bringt. Der BioTracker soll als schon kompiliertes Programm zur Verfügung stehen, sodass Nutzer dieses nur herunterladen müssen und sofort starten können.

A Anhang

A.1 Datenanalyse

Tabelle A.1: *Durchschnittlicher Positionsabstand und seine Standardabweichung in Abhängigkeit zu den Trackerparametern. Im 2. Teil der Tabelle sind Konfigurationen aufgeführt, bei denen das Tracking vorzeitig beendet wurde (nachdem sich die Bounding Box komplett von der zu verfolgenden Biene entfernt hat). Die Positionsabstände wurden in Relation zu der Diagonalen der Bounding Box gestellt. Dies ermöglicht den Vergleich dieser Werte mit Werten für andere Videos, auch wenn in diesen die Bienen größer oder kleiner sind.*

Korrekturbilder	Bilder bis Korrektur	Anzahl Features	realtiver durch. Positionsabstand	realtive Stdabw. Positionsabstand
10	10	5000	0.14	0.07
10	10	5100	1.29	0.68
10	10	10000	0.80	0.47
10	15	1000	0.15	0.08
10	17	1000	0.15	0.08
10	50	1000	2.15	1.01
10	50	5000	1.70	0.74
50	50	1000	0.95	0.79
50	50	5000	2.21	0.94
5	5	5000	1.17	3.20
10	10	1000	0.57	0.53
10	10	4500	0.37	0.34
10	10	5500	0.64	0.42

Tabelle A.2: *Durchschnittliche Ausrichtungsdifferenz und ihre Standardabweichung in Abhängigkeit zu den Trackerparametern. Im 2. Teil der Tabelle sind Konfigurationen aufgeführt, bei denen das Tracking vorzeitig beendet wurde (nachdem sich die Bounding Box komplett von der zu verfolgenden Biene entfernt hat).*

Korrekturbilder	Bilder bis Korrektur	Anzahl Features	durch. Differenz Ausrichtung [°]	Stdabw. Differenz Ausrichtung [°]
10	10	5000	86.16	50.28
10	10	5100	53.63	38.87
10	10	10000	55.58	40.64
10	15	1000	44.22	47.31
10	17	1000	18.06	14.94
10	50	1000	54.68	40.11
10	50	5000	63.67	45.45
50	50	1000	42.55	37.72
50	50	5000	57.13	40.19
5	5	5000	58.88	46.41
10	10	1000	72.85	46.20
10	10	4500	26.34	20.09
10	10	5500	47.73	38.12

Tabelle A.3: *Die verschiedenen Parameterkonfigurationen, mit denen das Video im SimpleTracker getracked wurde.*

Nummer Konfiguration	Anzahl Objekte	durch. Geschw.	min. Kontourgröße	Anzahl Erosionen	Anzahl Dilationen
1	7	75	25	10	5
2	7	75	25	12	6
3	7	75	50	12	6
4	8	50	25	3	3
5	8	75	25	3	3
6	8	75	25	6	5
7	8	75	25	8	5
8	8	75	25	10	5
9	8	75	60	14	7
10	10	50	25	3	3
11	10	75	25	3	3
12	12	50	25	3	3
13	12	75	25	3	3

Tabelle A.4: *Richtige, falsche und fehlende Detektionen in den Ergebnissen der verschiedenen Konfigurationen beim Tracking mit dem SimpleTracker.*

Nummer Konfig.	Detektionen	richtige Detektionen	richtige Detektionen %	falsche Detektionen	falsche Detektionen %
1	7216	4759	0.85	2457	0.34
2	6757	4640	0.83	2117	0.31
3	6621	4616	0.82	2005	0.30
4	13133	3636	0.65	9497	0.72
5	11419	3702	0.66	7717	0.68
6	9160	4941	0.88	4219	0.46
7	8680	4989	0.89	3691	0.43
8	7913	4896	0.87	3017	0.38
9	6711	4620	0.83	2091	0.31
10	18132	4171	0.75	13961	0.77
11	15304	4222	0.75	11082	0.72
12	20727	4596	0.82	16131	0.78
13	18394	4586	0.82	13808	0.75

Tabelle A.5: *Durchschnittlicher euklidischer Abstand und durchschnittliche absolute Differenz der verschiedenen Konfigurationen beim Tracking mit dem SimpleTracker.*

Nummer Konfiguration	durchschnittlicher eukl. Abstand [px]	durchschnittlicher rel. eukl. Abstand	Differenz Ausrichtung [°]
1	14.43	0.11	73.00
2	14.62	0.11	73.21
3	14.20	0.10	73.02
4	16.17	0.12	66.36
5	16.60	0.12	67.81
6	15.69	0.12	71.47
7	15.54	0.11	71.69
8	14.56	0.11	72.69
9	14.71	0.11	74.02
10	16.46	0.12	67.60
11	16.44	0.12	68.22
12	16.27	0.12	67.73
13	16.12	0.12	67.45

Tabelle A.6: *Anzahl der Fragmente in den, aus den Ergebnissen des automatischen Trackings, zusammengesetzten Pfaden. Diese werden der Anzahl der richtigen Detektionen gegenübergestellt, womit sich die durchschnittliche Länge eines Fragmentes ergibt.*

Nummer Konfiguration	richtige Detektionen	Anzahl Fragmente	durch. Fragmentlänge
1	4759	509	9.35
2	4640	511	9.08
3	4616	491	9.40
4	3636	732	4.97
5	3702	650	5.70
6	4941	641	7.71
7	4989	628	7.94
8	4896	559	8.76
9	4620	512	9.02
10	4171	1001	4.17
11	4222	834	5.06
12	4596	1170	3.93
13	4586	950	4.83

Literatur

[1] T. Landgraf, “Biorobtics Lab Website”, 2017.
URL: *http://berlinbiorobotics.blog/.*

[2] J. A. Tanke, *BioTracker - an extensible computer vision framework* Magisterarb. (Freie Universität Berlin, 2016).

[3] K. Branson, A. A. Robie, J. Bender, P. Perona und M. H. Dickinson, “High-throughput ethomics in large groups of Drosophila”, Nature America (2009).

[4] K. Branson, “Ctrax Website”, 2017.
URL: *http://ctrax.sourceforge.net/.*

[5] A. Pérez-Escudero, J. Vicente-Page, R. C. Hinz, S. Arganda und G. G. de Polavieja, “idTracker: tracking individuals in a group by automatic identification of unmarked animals”, Nature Amrica (2014).

[6] T. Landgraf und R. Rojas, *Tracking honey bee dances from sparse optical flow fields* Techn. Ber. (Freie Universität Berlin, 2007).