Bachelorarbeit am Institut für Informatik der Freien Universität Berlin,

Arbeitsgruppe Künstliche Intelligenz und Robotik, Biorobotics Lab

Modellierung eines probabilistischen Verfahrens zum Tracken von Bienenpfaden und Analyse der zugrunde liegenden Daten

Ronja Deisel
Matrikelnummer: 4467310
ronja.deisel@fu-berlin.de

Betreuer: Dr. Tim Landgraf

Berlin, 07.07.2015

Zusammenfassung

Im Rahmen des Projekts BeesBook werden ID-Tags auf Bienen-Thoraxen durch Bildverarbeitung erkannt. Diese Detektionen sollen zu Tracks zusammengefügt werden, wobei Fehler der Dekodierung ausgeglichen werden müssen. Dafür wurden die Fehler der Dekodierungen gegenüber Ground-Truth-Daten analysiert, um systematische Fehler aufzudecken und bei der Track-Findung berücksichtigen zu können.
Des Weiteren wurde ein Modell entwickelt, das die probabilistische Bewertung eines Tracks und seine Zuordnung zu einer Tag-ID ermöglicht.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

07.07.2015

Ronja Deisel

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Bienen und ihr Sozialverhalten 1
1.2 Projekt Beesbook 1

2 Problemstellung **3**
2.1 Funktionsweise des Decoders 3
2.2 Probleme des Decoders 4
2.3 Temporales Filtering 5

3 Modellierung und Analyse eines Temporalen Filters **6**
3.1 Modellierung 6
3.2 Implementierung und Ergebnisse auf generierten Daten 6
3.3 Implementierung und Ergebnisse auf echten Daten 8

4 Analyse der Fehler des Decoder-Programms **10**
4.1 Datenmengen und allgemeine Analyse 10
4.2 Häufigkeiten detektierter IDs 11
4.3 Hamming-Abstände 13
4.4 Bitflip-Positionen 14
4.5 Bitflip-Häufigkeiten des mittleren Bits von 3-Bit-Pattern 15
4.6 Zusammenfassung der Ergebnisse 16

5 Diskussion und Ausblick **17**

1 Einleitung

1.1 Bienen und ihr Sozialverhalten

In den letzten Jahren ist die Problematik der sterbenden Honigbienen (*Apis Mellifera*) weltweit immer mehr in das Bewusstsein der Öffentlichkeit geraten. Mehr und mehr Menschen betreiben in ihren Hinterhöfen Imkerei als anspruchsvolles aber ergiebiges Hobby. Dabei geht es nicht nur um das Interesse am Insekt und den Appetit auf Honig, sondern um die Erhaltung unseres Ökosystems. Denn die meisten Wild- und Nutzpflanzen sind auf die Bestäubung durch die Bienen angewiesen.[5]
Je mehr Bienen am Völkerkollaps – ausgelöst durch Milben, Pestizide, Monokulturhaltung u. a. – zugrunde gehen, umso wichtiger wird es, die Bienen und ihre Verhaltensweisen kennenzulernen und zu verstehen.

Honigbienen haben ein ausgeprägtes Sozialverhalten. Sie leben in Bienenstaaten mit Tausenden von Tieren zusammen und übernehmen wichtige Aufgaben im Staat. In bestimmten Zeiten begattet die Minderheit der Drohnen die Königin. Die Arbeiterinnen bauen die Waben, füttern Königin und Brut, reinigen den Stock und suchen nach Wasser und Nahrung. Hat eine Arbeiterin eine Futterstelle gefunden, fliegt sie zurück in den Stock und verkündet deren Ort, je nach Entfernung durch einen Rund- oder Schwänzeltanz. [3]

Die Bedeutung dieser Tänze ist inzwischen ausgiebig erforscht (vgl. u. a. [7]). Weniger untersucht wurde bisher aber eine weitere Eigenart der Bienen: Versuche mit der Roboterbiene Robobee [2] lassen vermuten, dass Bienen vorzugsweise in Cliquen kommunizieren. Dieser Vermutung versucht das Projekt Beesbook auf den Grund zu gehen.

1.2 Projekt Beesbook

Seit mehreren Jahren beschäftigt sich das Biorobotics Lab der Arbeitsgruppe Künstliche Intelligenz und Robotik der Freien Universität Berlin mit dem Kommunikationsverhalten der Bienen. In zahlreichen Abschlussarbeiten wurde im Projekt Beesbook versucht, die Überwachung und Analyse des Sozialverhaltens eines Volkes zu automatisieren.

In einem kühl, ruhig und dunkel gehaltenen Raum wurde ein Bienenstock mit bis zu 2000 Bienen aufgestellt. Ein Großteil der Bienen im Stock wurde einzeln per Hand mit einem QR-Code-ähnlichen Binär-Code beklebt, der der Biene eine ID zwischen 0 und 4095 zuteilt. Der Stock wurde im Sommer 2014 Tag und Nacht von beiden Seiten mit infrarotem Licht bestrahlt, da dieses die Bienen am wenigsten stört, dabei aber einfach durch Kameras abgelesen werden kann. Zusätzlich wurde jede Seite mit je zwei hochauflösenden (4000×3000 px) Kameras niedriger Frequenz (4 Hz) und einer hochfrequenten (100 Hz) Kamera niedriger Auflösung (160×120 px)

aufgenommen. Verschiedene Aspekte des Verhaltens werden durch verschiedene Programme getrackt, wobei die Hauptkomponenten für Folgendes zuständig sind:

- Die Waggle-Dance-Detection findet Schwänzeltänze in Echtzeit über die hochfrequente Kamera. [6]
- Der Decoder sucht auf den hochauflösenden Bildern nach Tags und speichert deren ID, Position und Drehwinkel ab .[4] [1]
- Andere Skripte überwachen die aufwendigen Aufnahme- und Berechnungsprozeduren, sichern Dateien und organisieren Daten in Datenbanken. [8]

In Zukunft sollen die Daten des Decoders durch Temporales Filtering zu Tracks zusammengefügt werden.

2 Problemstellung

2.1 Funktionsweise des Decoders

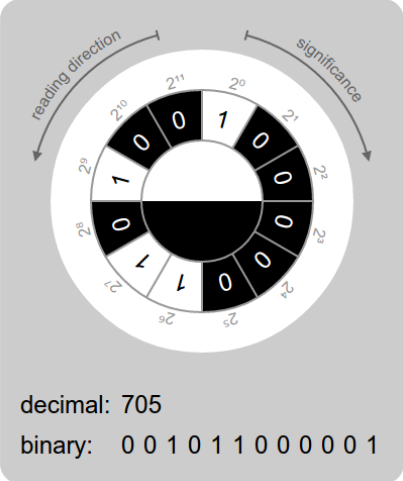


Abbildung 1: Tag-Design für ID 705

Das Innere des Tag-Designs besteht aus einem kleinen Kreis mit einer weißen und einer schwarzen Hälfte. Von der Mitte der weißen Hälfte aus im Uhrzeigersinn laufen zwölf Bits mit zunehmender Signifikanz als Segmente eines Kreisrings außen entlang des inneren Kreises, wobei schwarze Felder *FALSE* und weiße *TRUE* bedeuten (vgl. Abbildung 1). Die Tags wurden aus Papier mit einer dünnen Plastikbeschichtung ausgestanzt, wobei ein Abstand zum informationstragenden Kreisring eingehalten wurde. Insgesamt beträgt die Größe des Tags lediglich 3 mm und wiegt wenige Milligramm. Der Tag wird beim Stanzen leicht gewölbt und dann auf den Thorax einer Biene geklebt, sodass der weiße Halbkreis in Richtung des Kopfes liegt.

Der Decoder-Prozess besteht (zur Zeit der Auswertung der Bilder aus der Saison 2014) aus fünf Pipeline-Stufen:

- Der Preprocessor optimiert das Bild auf verschiedene Arten, um es auf die weiteren Schritte vorzubereiten, und führt dann eine Kantendetektion mit einem Sobel-Filter aus.
- Der Localizer führt eine Binarisierung gefolgt von einer Kombination der morphologischen Operationen Dilation und Erosion auf dem Sobel-Bild aus, wodurch auf dem schwarz-weißen Bild sogenannten 'Blobs' entstehen: einheitlich weiße runde Flecken, die nach Form und Größe einen Tag darstellen könnten. Für die Kandidaten werden umrahmende Bounding Boxes berechnet und an die nächste Schicht weitergegeben.
- Der Ellipse Fitter nimmt alle Regions of Interests des Localizers entgegen und sucht nach Ellipsen in den Kantenbildern des Bildabschnitts. Alle gefundenen Ellipsen werden nach ihrer Form bewertet. Für jede dieser "Regions of Interest" werden maximal die besten drei Ellipsen gespeichert, sofern deren Score eine Mindestqualität überschreitet.
- Der Grid Fitter versucht ein Raster eines Tag-Umrisses auf die Ellipse zu legen, dessen Kanten mit denen des darunterliegenden Tags übereinstimmen (siehe Abbildung 2). Die beiden Halbkreise in der Mitte dienen dabei hauptsächlich als Orientierung. Jeder Ellipsenkandidat wird wieder bewertet, und es werden

maximal die drei besten Kandidaten gespeichert, sodass nun insgesamt neun Kandidaten pro Region of Interest existieren können.

- Der Decoder vergleicht die durchschnittliche Helligkeit jeder durch das Grid vorgegebenen Zelle mit der der beiden inneren Halbkreise, um zu bestimmen, ob es sich um ein weißes oder schwarzes Feld handelt.

Für jede Region of Interest speichert der Decoder folgende Daten für jeden Ellipsen-Kandidaten in einer csv-Datei pro Bild: ihre Position, alle drei Winkel im Raum, ihre binäre ID und den Score des Grids, sowie die relevanten Daten des zugehörigen Grid-Kandidaten.

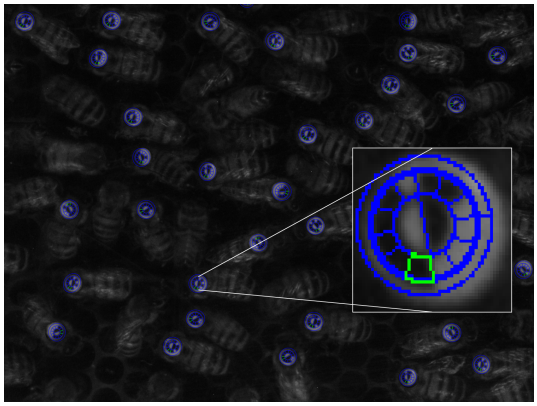


Abbildung 2: Jeder Tag-Kandidat wird durch ein Grid repräsentiert

2.2 Probleme des Decoders

Der Decoder ist eine große Verbesserung zum manuellen Tracken und macht die Analyse vieler Aspekte des Bienenverhaltens erst möglich. Dennoch hat er einige Probleme: schlechte Lichtverhältnisse, Spiegelungen an der Glasabdeckung des Stocks und der Plastikschicht des Tags, Verdeckung des Tags durch Verunreinigungen im Glas oder durch Flügel benachbarter Bienen, und sich in Bewegung oder in schräger Haltung befindliche Bienen verwirren alle Stufen der Detektion und machen die Suche nach passenden Ellipsen und Grids schwer (vgl. Abbildung 3). Das führt zu nicht oder falsch gefundenen Tag-Kandidaten und Grids, zu Bitdrehern und rotierten IDs. Es kommt zu unerwarteten Ausreißern, aber auch zu systematischen Fehlern.

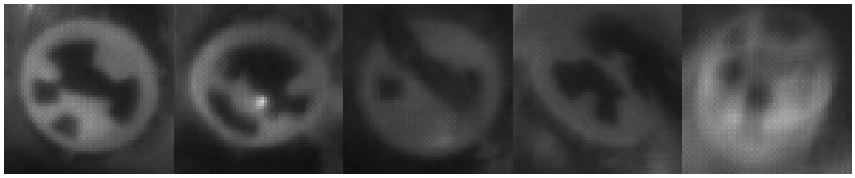


Abbildung 3: Auswahl von Tags mit unterschiedlicher Problemstellung demonstriert Fehlerquellen an Extremfällen: v.l.n.r.: gut lesbar, Spiegelung, teilweise durch Bienenbein überdeckt, stark gekippt, in Bewegung

Um deren Ausmaße einschätzen zu können und bei der Benutzung der Daten einberechnen zu können, habe ich im zweiten Teil dieser Arbeit die Art und Häufigkeit der Fehler genauer untersucht und versucht, deren Ursachen zu finden und Gegenmaßnahmen zu entwickeln.

2.3 Temporales Filtering

Unser Ziel ist, das Sozialverhalten der Bienen genauer zu analysieren. Das bedeutet Gruppen innerhalb des Volkes zu finden, die Effizienz der Tänze zu bewerten und die Bewegungsmuster der Arbeiterinnen zu studieren. Dazu benötigen wir, wenn möglich, den Aufenthaltsort jeder detektierten Biene im zeitlichen Verlauf. Die Lücken und Fehler des Decoders sollen dabei so gut es geht überbrückt werden.

Im ersten Teil dieser Arbeit habe ich deshalb ein Modell entworfen, das für eine Abfolge von detektierten IDs (inklusive deren Fehler) die Ground-Truth-ID findet, die wir unter den gegebenen Bedingungen für am wahrscheinlichsten halten. Wir gehen dabei davon aus, dass wir bereits mehrere solcher Tracks generiert haben und diese nur noch bewerten und einer ID zuweisen müssen.

3 Modellierung und Analyse eines Temporalen Filters

3.1 Modellierung

Wir benötigen eine Methode, um den optimalen aus mehreren möglichen Tracks auszuwählen und diesen einer ID und damit einer Biene zuordnen zu können. Eine erste Idee für eine solche Methode nennen wir Temporales Filtern.

Das Prinzip ist folgendes: wir betrachten einen einzelnen Track unter der Annahme, dass alle vom Decoder identifizierten IDs auch wirklich zu dem Track gehören. Entstehen kann ein solcher Track, indem Detektionen zusammengefügt werden, die sich zwischen zwei Bildern örtlich nicht weit voneinander entfernt haben. Wir nehmen außerdem bestimmte Wahrscheinlichkeiten für das Auftreten von n Bitflips an. Unter Bitflips verstehen wir ein Bit, das von 0 nach 1 oder von 1 nach 0 kippt. Diese Wahrscheinlichkeiten werden in eine Tabelle (im Folgenden Probability Table genannt) eingetragen, die beschreibt, mit welcher Wahrscheinlichkeit es sich bei jeder möglichen ID um die Ground-Truth-ID handelt, wenn eine bestimmte ID detektiert wurde. Wählt man für jede gelesene ID im Track die entsprechende Zeile der Probability Table aus und multipliziert all diese mit Punktmultiplikation miteinander, dann erhält man eine Einschätzung über die Wahrscheinlichkeit jeder ID als Ground-Truth-ID. Die Spalte mit dem größten Wert enthält die wahrscheinlichste ID. Im Grunde handelt es sich dabei um eine Markov-Kette hoher Dimension.

Im Rahmen dieser Arbeit habe ich dieses Modell entworfen, um herauszufinden, wie gut es bei den uns zugänglichen Daten funktioniert, wie viele Bilder benötigt werden, bis das Modell eine genügend sichere Vermutung anstellen kann, und dadurch abwägen zu können, ob es Sinn macht ein solches System in der Praxis anzuwenden.

3.2 Implementierung und Ergebnisse auf generierten Daten

In der Ausgangssituation der Problemstellung war der Decoder noch nicht großflächig im Einsatz. Es existierten noch keine benutzbaren Daten und lediglich ungenaue Vorstellungen davon, von welcher Qualität diese sein würden. Fehler-Wahrscheinlichkeiten wurden deshalb nach bestem Ermessen geschätzt.

Im Zuge der Modellierung entstand ein Python-Skript, das das Modell im Kleinen implementiert und testet.
Dafür wurden Tracks generiert, von denen eine bestimmte Anzahl an IDs mit Bitdrehern versehen wurden. Die Wahrscheinlichkeiten der Bitdreher richteten sich nach derselben Probability Table, die später zum Filtern verwendet wurde. Zusätzlich wurden Tracks mit den gleichen Voraussetzungen generiert, bei denen aber die maximale Anzahl der Bitdreher pro ID festgesetzt war.
Dann wählt das Skript für den Track die entsprechenden Zeilen aus, multipliziert diese miteinander und gibt das Maximum aus.

Dieser Vorgang wurde für verschiedene Tag-Längen, Track-Längen, Fehler-Wahrscheinlichkeiten sowie maximale Anzahlen an Bitdrehern häufig wiederholt, wobei die Qualität der Ergebnisse gespeichert wurde. Diese wurden in einem zusätzlichen Python-Skript ausgewertet und geplottet.

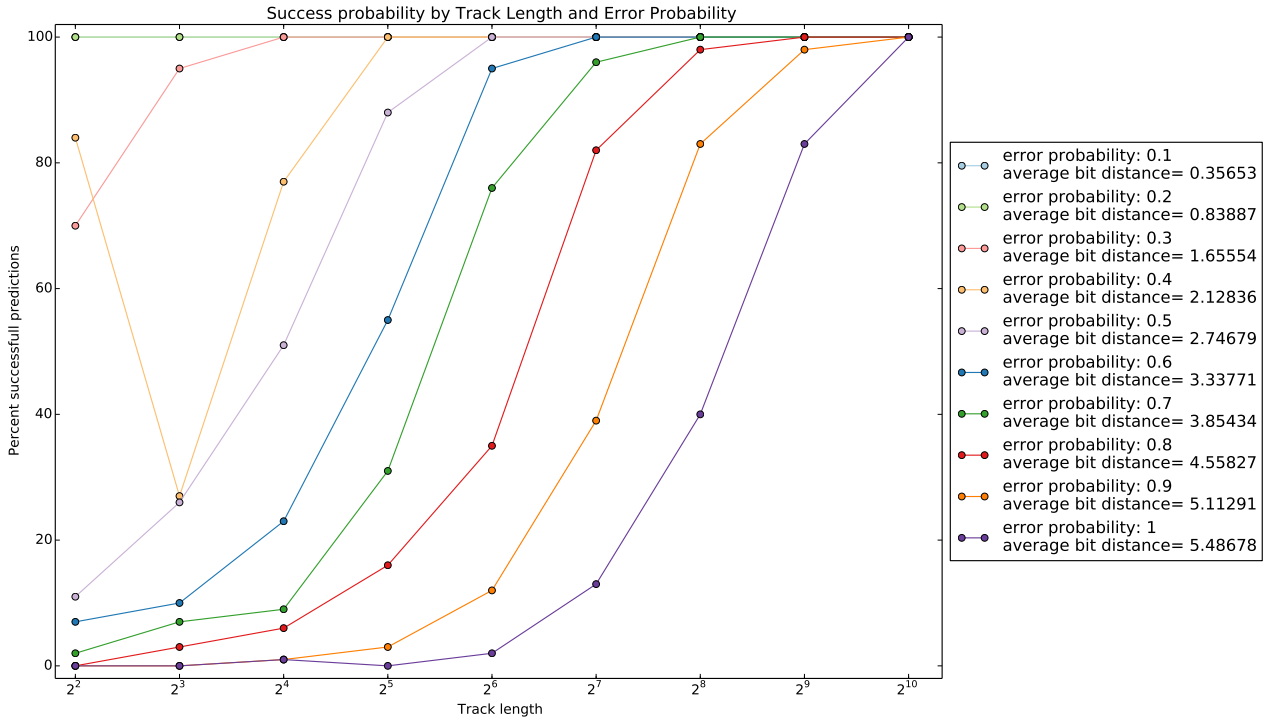


Abbildung 4: Plots der Performance-Tests des Temporalen Filtering an generierten Daten

Es hat sich herausgestellt, dass die Modellierung mit einer eingeschränkten Bitflip Anzahl die Realität nicht widerspiegelt, weshalb hier nur der Fall ohne Einschränkung der Bitflip-Anzahl besprochen wird.
In Abbildung 4 ist dieser Fall geplottet: Er zeigt den Anteil korrekt vorausgesagter IDs abhängig von der Track-Länge sowie der Error Probability, also der Wahrscheinlichkeit einer Störung in der Detektion. Zusätzlich wird der durchschnittliche Hamming-Abstand aller IDs zu jeder Error Probability angegeben.
Da wir in späteren Experimenten einen durchschnittlichen Hamming-Abstand von $1,87$ und die Wahrscheinlichkeit einer komplett korrekten ID von 24% erhalten werden (siehe Abschnitt 4.3), scheint der am besten passende Wert irgendwo zwischen Error Probability $0,5$ und $0,8$ zu liegen. Wie bereits erwähnt, ist die Methode für die Generierung eines gestörten Tracks leider kaum möglich, weshalb wir uns hier mit sehr grob geschätzten Werten zufrieden geben müssen.
Trotzdem erkennt man folgendes aus dem Plot: Mit unserer ungefähren Error Probability von 0,7 ist eine Tracklänge von ca. $2^7 = 128$ nötig, um eine ausreichende Qualität ($> 90\%$) zu liefern. Bei vier Bildern pro Minute und ca. $0,78$ Detektionen

pro erwarteter Detektion (auch dieser Wert stammt aus einer späteren Analyse, siehe Abschnitt 4.3), benötigen wir so also etwa $\frac{128}{0{,}78 \cdot 4} = 41$ Sekunden an Material. Das wäre zwar mehr als erhofft, läge aber noch in einem vertretbaren Rahmen.

3.3 Implementierung und Ergebnisse auf echten Daten

Nachdem der Decoder im Laufe der Arbeit für eine Zeitlang im Einsatz war und Ergebnisse produzieren konnte sowie gleichzeitig Ground-Truth-Tracks für einige IDs per Hand erstellt wurden, konnte ein weiteres Experiment gestartet werden: Hier wurden die echten Decoder-Ergebnisse (statt der generierten) eingelesen, durch das Model geschleust und die Anzahl der korrekt vorhergesagten IDs gemessen, wobei verschieden lange Teilstücke aus verschiedenen Untermengen des Tracks zur Berechnung herangezogen wurden.

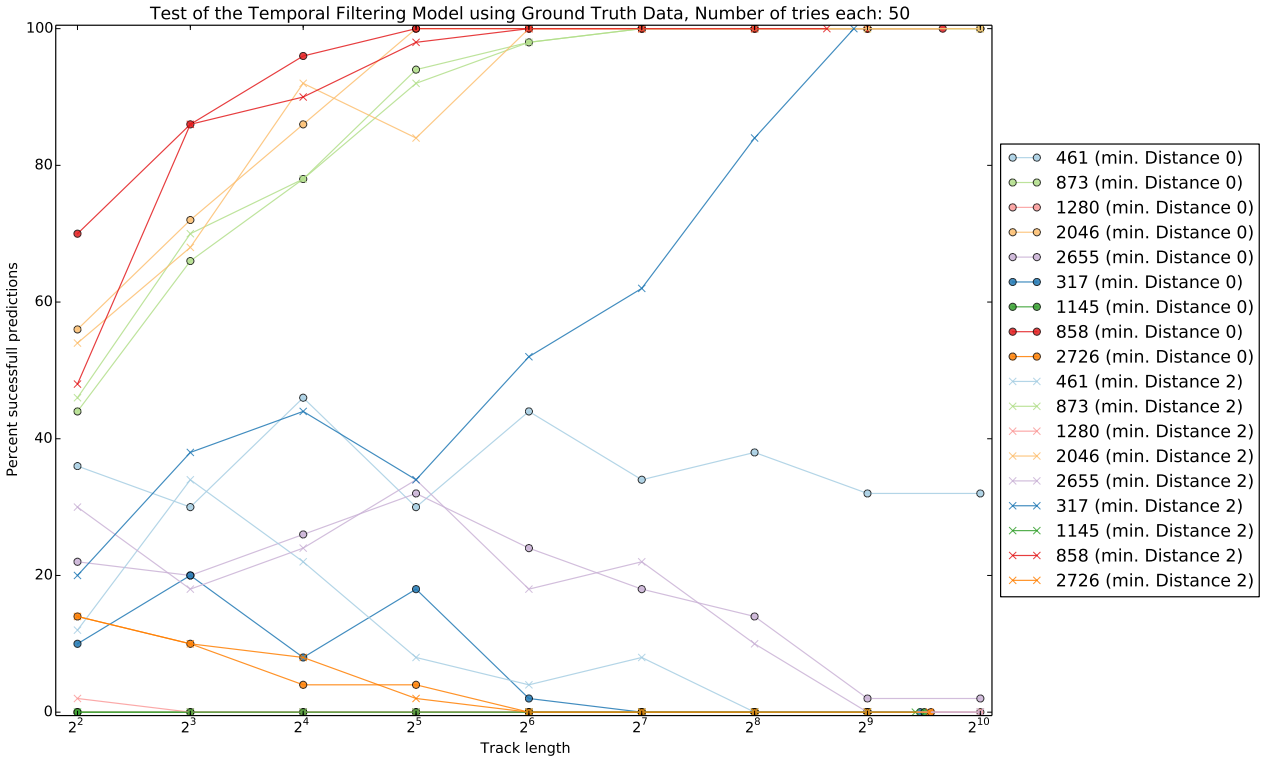


Abbildung 5: Plots der Performance-Tests des Temporalen Filtering an echten Daten

Dieser Versuch wurde einmal mit den originalen Tracks ausgeführt und einmal mit einem gefilterten Track, der nur IDs beinhaltet, die sich zwischen zwei Bildern um mehr als 2 Pixel (nach euklidschem Abstand) bewegt hatten. Beweggrund dieser zusätzlichen Untersuchung war die Annahme, dass Detektionen, die sich sowohl zeitlich als auch örtlich kaum verändert hatten, unter sehr ähnlichen Bedingungen aufgenommen wurden und deshalb wenig neue Informationen liefern konnten. Wenn

sich diese Annahme bestätigt, würde das also heißen, dass derartige Berechnungen abgekürzt werden können.

Abbildung 5 visualisiert die Ergebnisse für die neun längsten Ground-Truth-Tracks. Man sieht, dass manche Tracks mit größerer Track-Länge gegen eine falsche ID konvergieren, andere gegen eine richtige. Die meisten Tracks konvergieren zwischen Track Länge $2^4 = 16$ und $2^7 = 128$. Da in diesen Daten leere Detektionen bereits mit eingerechnet sind, fallen diese Ergebnisse besser aus, als die Tests mit den generierten Daten erhoffen ließen.
Betrachtet man die Unterschiede in den gefilterten und nicht gefilterten Daten, sieht man, dass die gefilterten Tracks in der Regel schneller konvergieren, das Ergebnis aber – außer in einem Fall – nicht verändern. In dieser einen Ausnahme (ID 317) konvergiert der gefilterte Track gegen die korrekte ID, der ungefilterte aber nicht. Es wäre also sinnvoll nur Detektionen zu betrachten, die sich zwischen zwei Bildern nennenswert bewegt haben, da dies im schlechtesten Fall nur schneller zu Ergebnissen führen, im besten Fall aber sogar die Qualität des Ergebnisses verbessern kann.

Das Temporale Filtering kann also – abhängig von der Qualität der Eingangsdaten – gute und schnelle Ergebnisse liefern.

4 Analyse der Fehler des Decoder-Programms

Während der Entwicklung des Decoders wurde für lange Zeit der Hamming-Abstand zur Ground-Truth (also die Anzahl der unterschiedlichen Positionen der Strings) als einziges Güte-Merkmal für die Detektionsqualität angenommen. Dabei fielen immer wieder Fehler auf, die anderen Regeln zu folgen schienen und deshalb nicht alleine durch den Hamming-Abstand erklärt werden können.
Deshalb entschieden wir uns, die Fehler des Decoders nach folgenden Aspekten genauer zu analysieren:

- Häufigkeiten einzelner IDs
- Hamming-Abstände
- Positionen der Bitflips
- Verhalten des mittleren Bits von 3-Bit-Patterns

Zu diesem Zweck habe ich ein Python-Skript implementiert, das die Analyse der Aspekte ausführt und die Ergebnisse in schriftlicher sowie graphischer Form ausgibt bzw abspeichert.

Zuerst mussten Ground-Truth-Tracks für verschiedene IDs erstellt werden. Dafür wurde ein bereits im Projekt existentes MATLAB-Skript verwendet, mit dem durch einfache Maus-Klicks eine einzelne Biene über eine große Menge von Bildern verfolgt werden kann. Das Skript speichert für jeden Klick den Pfad der verwendeten Bilddatei und die Position im Bild, und ordnet dem Klick die vom Decoder gefundene Detektion zu, die dem Klick in einem kleinen Umkreis am nähesten ist. Die ID-Nummer, die der Detektor dieser Detektion zugeteilt hat (dies entspricht nicht der detektierten ID sondern dient nur der Identifikation in den Ausgabedateien des Decoders) wird ebenfalls zum Klick gespeichert.
Getrackt wurden fünf IDs (davon eine Tänzerin) über einen Zeitraum von etwa 54 Minuten, solange sich diese im Aufnahmebereich der Kameras befanden. Zusätzlich wurden durch eine ebenfalls schon existierende Python-GUI 22 Bienen über etwa 500 Bilder getrackt um IDs in der Breite abzudecken, wobei hier nur für detektierte IDs Ground-Truths gespeichert wurden.
Im Folgenden werde ich Vorgehen, Ergebnisse und Schlussfolgerung aller analysierten Aspekte beschreiben.

4.1 Datenmengen und allgemeine Analyse

Zuallererst war es wichtig einen Überblick über die Datenmenge und -verteilung zu bekommen.
Dafür wurde die Anzahl der per Hand getätigten Markierungen, der für diese Markierungen gefundenen Detektionen und der für diese Detektionen gefundenen ID-Kandidaten aufgezeichnet. Tabelle 1 zeigt diese Daten für die fünf längsten

Markierungstracks sowie die Gesamtanzahl für die Summe aller Markierungen, sowohl für den ungefilterten Fall als auch den Fall, in dem alle Detektionen zwischen Bildern einen euklidschen Abstand von mindestens 2 Pixeln haben:

G.-T.-ID	G.-T.-ID(bin.)	Markierungen		Detektionen		Kandidaten	
Filter	-	0	2	0	2	0	2
461	000111001101	6094	4585	4465	2956	13722	9162
873	001101101001	1024	949	878	803	2943	2688
1280	010100000000	4608	3285	3678	2355	12675	8133
2046	011111111110	2304	1846	1668	1210	5664	4008
2655*	101001011111	3309	3191	2309	2191	8001	7587
alle	-	19962	16193	15621	11852	51663	39309

* Tänzerin

Tabelle 1: Anzahl Markierungen, Detektionen und ID-Kandidaten pro Ground-Truth-ID (G.-T.-ID), ungefiltert und gefiltert nach Mindestabstand 2

Betrachten wir zuerst den ungefilterten Fall:
Durchschnittlich werden hier für 78% aller Markierungen Detektionen gefunden, bei der tanzenden Biene fällt das auf ein Minimum von 70% ab. Dieser Qualitätsunterschied ist zu erwarten, da sich eine tanzende Biene in Bewegung befindet und oft nur unscharf aufgenommen werden kann.
Durch die Filterung werden etwa 19% aller Markierungen verworfen. Das hat zur Folge, dass die Anzahl der Detektionen pro Markierung auf 73% schrumpft, wir verlieren also 5% der Daten.

4.2 Häufigkeiten detektierter IDs

Um mir einen Überblick über die Verteilung der Daten zu verschaffen und erste Hypothesen aufzustellen, bestand die nächste Analyse aus einem einfachen Histogramm der detektierten IDs (getrennt nach Ground-Truth-ID). Für einen weiteren Informationsgewinn betrachtete ich diese in Abhängigkeit zu ihrem Hamming-Abstand zur Ground-Truth.
Abbildung 6 zeigt zwei beispielhafte resultierende Plots, wobei in diesen (um Übersichtlichkeit zu bewahren) nur maximal 30 der häufigsten IDs gezeigt werden. Auf der x-Achse sind die detektierten IDs lexikographisch sortiert gegen die insgesamte Detektionshäufigkeit auf der logarithmischen y-Achse aufgetragen. Die Balkenfarbe hängt von der Hamming-Distanz der detektierten ID zur Ground-Truth ab. Die Ground-Truth selber ist dabei dunkelgrün eingefärbt.
Die gezeigten Plots sind typische Beispiele für Tracks bei ausreichender Datenmenge.
Erwartungsgemäß sind in beiden Plots hauptsächlich grüne – also der Ground-Truth ähnliche – IDs zu sehen. Die korrekt detektierte ID ist für Ground-Truth 461 mit einem Unterschied von mehr ca. 31% die häufigste, steht bei ID 2655 aber mit

12% Unterschied nur an zweiter Stelle. Hier ist anzunehmen, dass die Bewegung der tanzenden Biene einen “Motion Blur”, also das Verschwimmen von Features, im Bild verursachte, und ein Pattern von ≪101≫ wie ≪111≫ aussehen ließ.

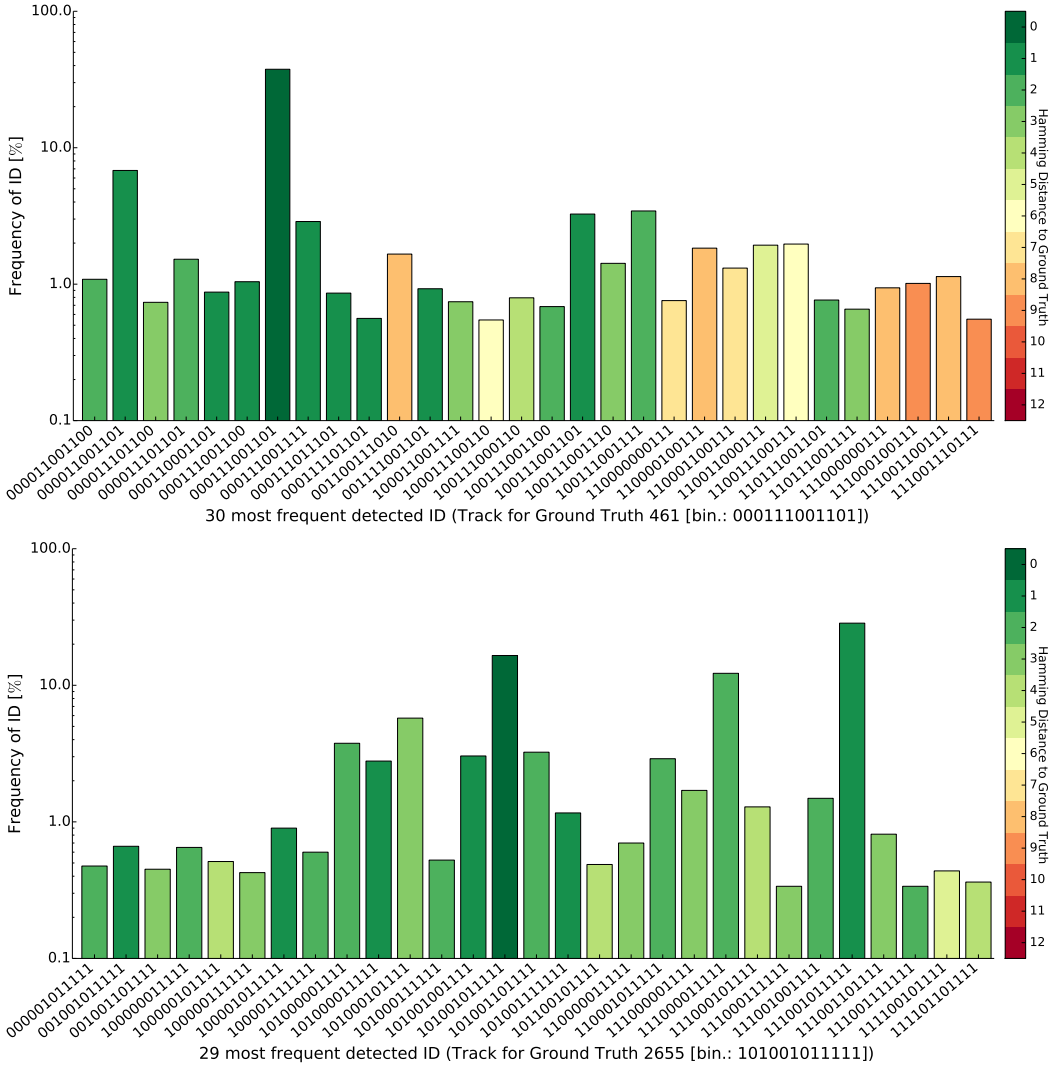


Abbildung 6: Beispielhafte Plots der Häufigkeiten der detektierten IDs für zwei Ground-Truth-IDs: 461 und 2655 (Tänzerin)

Augenscheinlich kommt es häufig vor, dass Löcher in Flächen geschlossen werden, also einzelne schwarze oder weiße Bits flippen und von ihrer Umgebung überlagert werden. Außerdem scheinen manche Muster unabhängig von der Ground-Truth ID häufiger als andere vorzukommen, nämlich z. B. ID 3591 (schwarzer Halbkreis auf der schwarzen Tag-Hälfte) und IDs 3687, 3815, 3831, 3703, die sich alle als schwarze Linie interpretieren lassen und eventuell auch auf falsch positionierten Detektionen auf den Streifen des Hinterleibs der Biene gelesen werden können.

Die erstgenannte Hypothese, dass Motion Blur und andere Unschärfe zum Schließen von Löchern und Verschieben von Linien führt, wird hier nur beobachtet und im Verlauf dieser Arbeit genauer untersucht und quantifiziert. Die Hypothese der insgesamt häufig vorkommenden IDs kann leider im Rahmen einer Untersuchung mit ID-Tracks für nur 27 Tiere nicht ausreichend quantifiziert werden und bleibt daher nur eine interessante Vermutung.

4.3 Hamming-Abstände

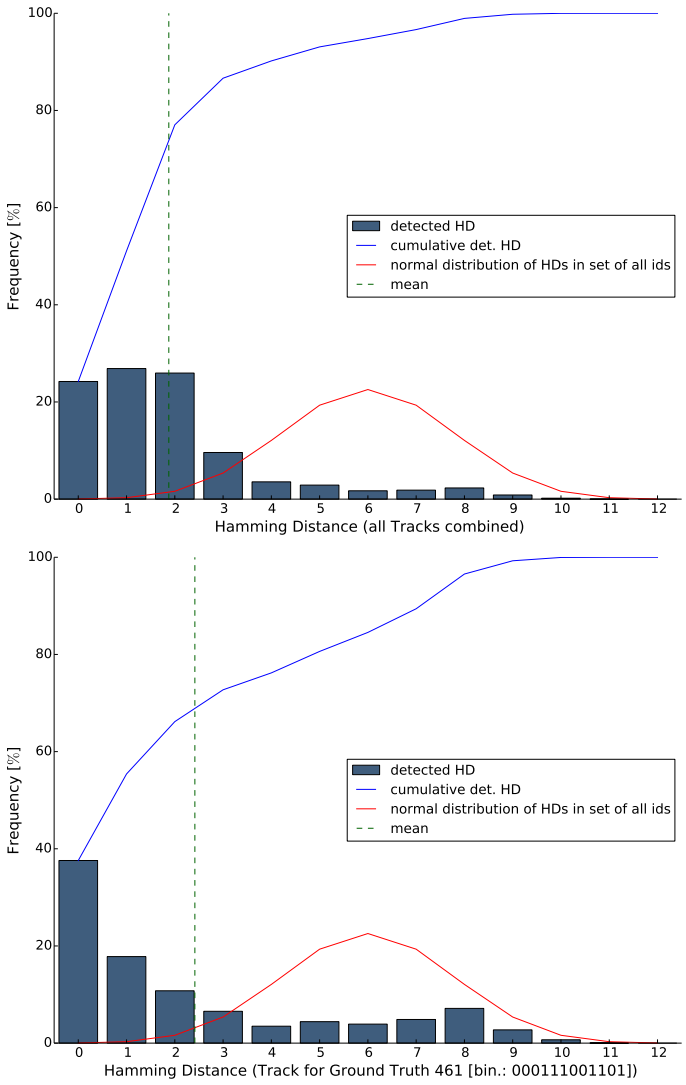


Abbildung 7: Beispielhafte Plots der Häufigkeiten der Hamming-Abstände zur Ground-Truth für die Kombination aller Tracks sowie die ID 461

In einem weiteren Plot wurden die Häufigkeiten der Hamming-Abstände 0 bis 12 zur Ground-Truth aufgezeigt. Diese Art von Plot wurde einmal für jeden einzelnen Track und einmal für alle Tracks zusammen genommen durchgeführt. Im gleichen Plot wurde der kumulative Hamming-Abstand, das arithmetische Mittel der Verteilung, sowie die Normalverteilung der Hamming-Abstände für die Menge aller (einzigartigen) IDs gezeichnet. Das dient einer besseren Einschätzung der Qualität des Tracks, da z. B. ein relativ hoher Balken bei Hamming-Distanz 1 akzeptabel sein kann, weil sich diese Häufigkeit auf 12 Kombinationen mit demselben Hamming-Abstand aufteilt.

Abbildung 7 zeigt die Analyse der Hamming-Abstände für eine Kombination aller Tracks sowie für ID 461. Das arithmetische Mittel der Hamming-Abstände beträgt für alle Tracks zusammen genommen 1,9.
Durch diesen Plot werden auf einen Blick verschiedene Qualitätsmerkmale der Tracks ersichtlich: Beispielsweise zeigt die Verteilung für ID 461, dass hier die richtige ID überaus häufig gefunden wurde, relativ

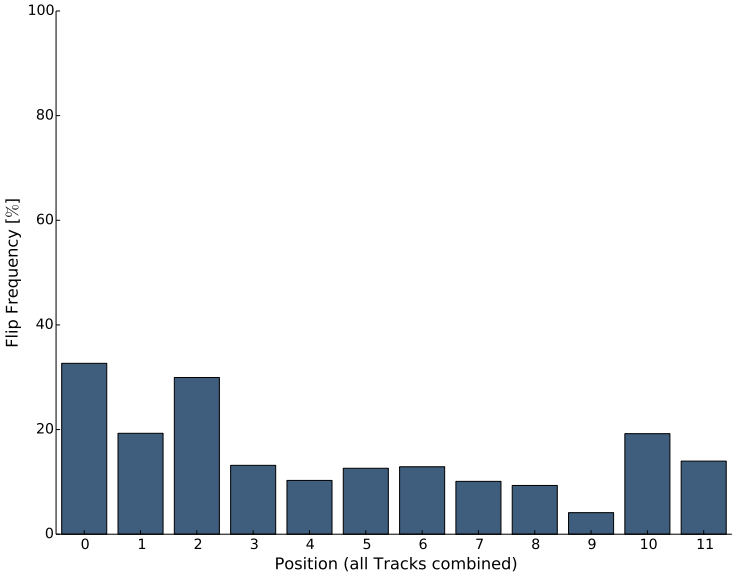
gute aber nicht perfekte IDs aber eher seltener im Vergleich zu dem Plot der Kombination aller IDs. Da dies aber eher die Ausnahme ist und im allgemeinen Fall 90% aller Detektionen einen Hamming-Abstand zwischen 0 und 3 haben, ist es sinnvoll, diesem doch eine starke Gewichtung bei der ID-Zuweisung zuzuordnen.

Der Unterschied zu der gefilterten Version ist zu gering, um daraus etwas schlussfolgern zu können.

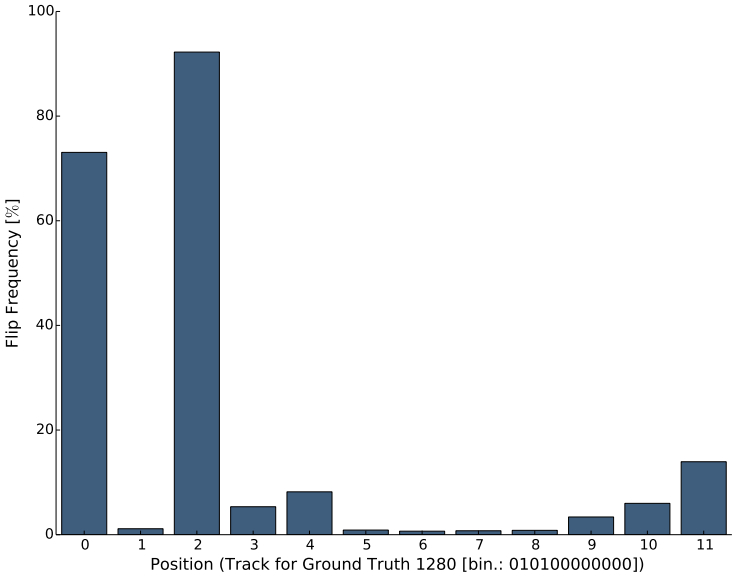
4.4 Bitflip-Positionen

Als nächstes wurde die Häufigkeiten von Bitdrehern nach Position auf dem Tag untersucht. Abbildung 8a zeigt das Ergebnis für die Kombination aller Tracks. Die Positionen wurden nach Leserichtung benannt, also entgegen dem Uhrzeigersinn in der Mitte der weißen Seite beginnend.

Es fallen die Maxima bei Position 0 und 2 mit Abständen größer als 10% zur nächstkleineren, sowie ein schwaches Minimum bei Position 9 auf. Da hierfür aber keine Erklärung einleuchtend erscheint, bleibt nur zu vermuten, dass es sich hierbei um ein Artefakt der Auswahl der Ground-Truth-IDs handelt.
Ein Blick auf die Plots der längsten Ground-Truth-Tracks bestätigt diese Annahme: Vor allem bei ID 1280 (siehe Abbildung 8b) häufen sich Bitflips in Position 0 und 2 mit Wahrscheinlichkeiten $> 70\%$ bzw $> 90\%$, was mit dem Aufbau des Tags zu tun hat: Im Tag befinden sich hier schwarze



(a) alle IDs



(b) ID 1280

Abbildung 8: Häufigkeiten von Bitflips nach Position auf dem Tag für alle Tracks und ID 1280

Bits, die durch die benachbarten weißen Bits überstrahlt werden.

Diese Art von Plots kann also hilfreich sein für das Erkennen von Problemen einzelner IDs oder ähnlicher Bitkonstellationen.

4.5 Bitflip-Häufigkeiten des mittleren Bits von 3-Bit-Pattern

Zuletzt wurde untersucht, wie sich die Flip-Häufigkeit auf verschiedene 3-Bit-Pattern verteilt, d. h. wie oft ein Bit kippt, abhängig von dem Muster, das die Ground-Truth neben und in diesem Bit zeigt. Die Erwartung, die sich in den vorangegangenen Analysen herauskristallisiert hatte, war, dass durch Unschärfe im Bild kleine Bereiche wie einzelne Felder im Tag-Design verschwimmen und nicht mehr korrekt gelesen werden können.
Zusätzlich dazu betrachtete ich diese Metrik hinsichtlich der Lage des mittleren Bits: wo befindet es sich, relativ zum schwarzen oder weißen Halbkreis?
Der Plot in Abbildung 9 visualisiert die Ergebnisse für alle IDs jeweils im Verhältnis zu der Häufigkeit der Muster.

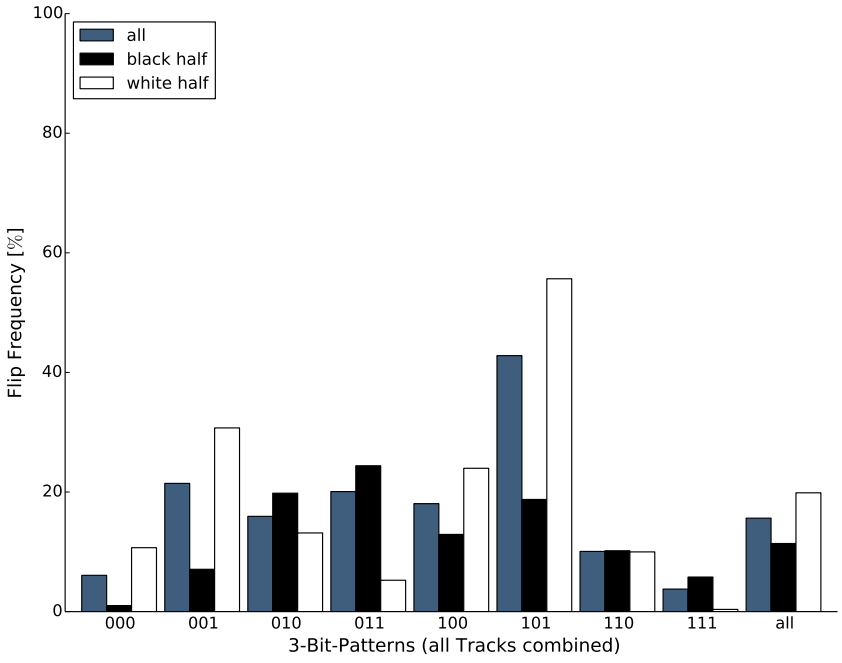


Abbildung 9: Häufigkeit für das Flippen eines mittleren Bits innerhalb verschiedenen Bitkombinationen (0 = schwarz, 1 = weiß)

Betrachten wir zuerst die Flip-Häufigkeit unabhängig davon, auf welcher Hälfte des Tags sich das Bit befindet: Deutlich sichtbar mit ca. 40% ist das Maximum bei der Kombination ≪101 ≫. Hier flippt ein schwarzes Bit, das sich zwischen zwei weißen befindet. Dies entspricht stark unserem erwarteten Ergebnis: durch verschwommene Grenzen ist ein einzelnes schwarzes Bit schwer erkennbar.

Im Kontrast dazu widerspricht die zweite Beobachtung unseren Erwartungen: Die Kombination ≪010 ≫ (ein isoliertes schwarzes Bit wird als weiß gelesen) ist, wenn beide Hälften

betrachtet werden, sehr unauffällig. Ein einzelnes weißes Bit ist also nicht fehleranfälliger als das durchschnittliche Bit.

Minima finden wir bei den Kombinationen ≪000 ≫ und ≪111 ≫, wo sich ein komplett schwarzes bzw weißes Feld gebildet hat.

Bei zusätzlicher Betrachtung der Tag-Hälfte, fallen Häufigkeitsunterschiede vor allem bei der Kombination ≪101≫ auf. Ein völlig isoliertes schwarzes Bit flippt mit einer Wahrscheinlichkeit von fast 60%. Das lässt auf eine eventuelle Übersteuerung der Beleuchtung oder eine zu starke Konvolution bei der Vorverarbeitung des Bildes schließen.

Zusätzlich ist bemerkenswert und auch einleuchtend, dass jedes schwarze Bit häufiger auf der weißen Hälfte und jedes weiße Bit häufiger auf der schwarzen Hälfte als auf der gegenüberliegenden flippt. Auch das spricht für das Verwischen von Grenzen durch Unschärfe.

Insgesamt erkennen wir auch – neben dem erwarteten Unschärfe-Effekten – ein kleines aber deutliches Bias zugunsten der schwarzen Tag-Hälfte, vermutlich, da dort sowohl eine schwarze (innerer Halbkreis) als auch eine weiße (äußerer Kreisring) Fläche als Anchoring dienen kann.

4.6 Zusammenfassung der Ergebnisse

Zu bedenken gibt es, dass es sich hier um eine Stichprobe von nur 27 unterschiedlichen IDs handelt, deren Bitverteilung ein Bias aufweisen können. Zusätzlich sind die IDs unterschiedlich stark repräsentiert. Gerade die Plots über die Position sind hierfür sehr anfällig und sollten deshalb nicht zu ernst genommen werden, solange sie nicht nur über einzelnen Tracks berechnet werden. Für isolierte Ground-Truth-IDs sagen die Plots viel aus, was aber nie auf alle Tracks übertragbar ist.

Da sich aber die Vermutung der Unschärfeeffekte in den meisten Plots bestätigt hat, muss dieses Problem in Zukunft genauer betrachtet werden. Lösungsansätze für dieses Problem enthalten u. a. bessere Hardware mit einer größeren Auflösung und Frequenz. Alternativ kann versucht werden, die erhaltenen Werte mit in die Beurteilung der Detektionsqualität einfließen zu lassen, sodass z. B. einer übereinstimmenden weißen Tag-Seite mehr Gewicht zugeteilt wird als einer schwarzen.

5 Diskussion und Ausblick

Ziel dieser Arbeit war es, die Fehler, die sich durch die Hardware und den Decoder-Code ergeben, einschätzen zu können. In Zukunft sollen dadurch Gegenmaßnahmen entwickelt werden. Einige der aufgedeckten Probleme könnten durch eine Nachrüstung der Software gelöst werden. Da es sich um ein lebendiges unvorhersehbares System handelt, ist es aber in anderen Fällen oft nicht möglich, genaue Muster in den Daten zu finden. Dafür wäre eine unrealistisch große Menge an Ground-Truth-Daten notwendig. Um aber speziellen Fragestellungen und Eigenheiten der Dekodierung nachzugehen, lassen sich die entstandenen Skripte gut nutzen.

Den größten Einfluss auf die Qualität der Ergebnisse eines Temporalen Filters hat immer der Decoder. Die Weiterentwicklung des Decoders sollte also auch in Zukunft oberste Priorität haben, da nur mit genauen Ergebnissen auch genaue Rückschlüsse möglich sind.

Ein weiterer Ansatzpunkt ist die Überlegung, dass Detektionen, deren Bedingungen sich wenig unterscheiden, redundant sind und aus der Berechnung herausgenommen werden können. Diese Hoffnung hat sich in Bezug auf die Position der Detektion bewahrheitet, könnte in Zukunft aber noch um viele Aspekte erweitert werden, wie z. B. den des Winkels, den der Beleuchtungsverhältnisse und den der Umgebungsbedingungen.

Literatur

[1] Herm, F. Automatische Detektierung von Bienentänzen, 2011.

[2] Landgraf, T., Rojas, R., Nguyen, H., Kriegel, F., and Stettin, K. Analysis of the Waggle Dance Motion of Honeybees for the Design of a Biomimetic Honeybee Robot. *PLoS ONE* (2011).

[3] Mackean, D. G., and Mackean, I. The Honey Bee (Apis mellifera), 2004-2015. [Online; accessed 7-July-2015].

[4] Michels, M. A Beehive Monitoring System Incorporating Optical Flow as a Source of Information, 2011.

[5] Oberschelp, L. Bienensterben — das Verschwinden der fleißigen Helfer, 2012. [Online; accessed 7-July-2015].

[6] Rau, A. Realtime Honey Bee Waggle Dance Decoding System, 2014.

[7] Riley, J. R., Greggers, U., Smith, A. D., Reynolds, D. R., and Menzel, R. The flight paths of honeybees recruited by the waggle dance. *Nature* (2005).

[8] Tietz, C. Entwurf und Implementierung einer Speicherarchitektur für Bildmassendaten zur Verhaltensanalyse von Honigbienenkolonien, 2015.