Bachelorarbeit am Institut für Informatik der Freien Universität Berlin,

Arbeitsgruppe Künstliche Intelligenz und Robotik, Biorobotics Lab

Optimierung von Binarisierungsverfahren auf Bildern von Barcodemarkierungen auf Bienen

Amjad Saadeh
Matrikelnummer: 4554300
amjad.saadeh@fu-berlin.de

Eingereicht bei: Prof. Dr. Raoul Rojas
Betreuer / Zweitgutachter: Tim Landgraf

Berlin, 22.07.2014

Amjad Saadeh

Zusammenfassung

Die Honigbiene ist eines der ältesten und wichtigsten Nutztiere der Menschheit. Obwohl das Sozialverhalten von Bienenvölkern gut erforscht ist, gibt es immer noch Aspekte ihres Lebens, die uns Rätsel aufgeben. Um diesen Aspekten auf den Grund zu gehen, entwickelt das Beesbook-Projekt ein System zum Identifizieren und Verfolgen von Bienen innerhalb ihres Stocks. Dieses System arbeitet mit Graustufenbildern und nutzt schwarz-weiße QR-Code ähnliche Markierungen auf dem Bienentorax, welche dekodiert werden müssen. Hierzu müssen schwarze und weiße Zellen zuverlässig unterschieden werden können.

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.

22.07.2014

Amjad Saadeh

Inhaltsverzeichnis

1 Die Honigbiene

Die Europäische Honigbiene (*Apis mellifera*) ist eines der ältesten Nutztiere der Menschheit. Bereits ca. 3000 vor Christus, zu den Zeiten des alten Ägypten, wurde Honig von Wildbienen geerntet [1].
Bis heute ist die Bedeutung der Honigbiene ungebrochen. Dabei hält sie nicht nur die offensichtlichen Rolle als Honigproduzent inne, sondern leistet auch einen wichtigen Beitrag zur Bestäubung von Pflanzen. So sind zahlreiche Nahrungs- und Futterpflanzen auf eine Fremdbestäubung durch Bienen oder artverwandte Insekten angewiesen [2].

1.1 Grundlegende Struktur eines Bienenvolkes

Ein Bienenvolk besteht in der Regel aus mehren tausend Individuen. Die meisten sind Arbeiterinnen, welche, je nach Alter [3], verschiedene Aufgaben wahrnehmen. So sind jüngere Arbeiterinnen eher für die Pflege der Brut und der Königin verantwortlich, wohingegen ältere Arbeiterinnen sich der Nahrungsbeschaffung widmen. Hinzu kommen noch einige männliche Bienen, die Drohnen, deren einzige Aufgabe die Begattung einer Jungkönigin ist. An der Spitze steht die Königin, welche hauptsächlich mit dem Legen neuer Eier beschäftigt ist.

1.2 Kommunikation

Zum Aufrechterhalten eines solch komplexen Konstrukts, wie einem Bienenvolk, bedarf es Kommunikationsmöglichkeiten. Ein Medium mit dem sich vor allem die Königin mitteilt sind Pheromone. Die von der Königin abgegeben Botenstoffe sind Teil des sog. *Königinnensekrets*, welches u.a. die Arbeiterinnen davon abhält neue Königinnen aufzuziehen oder ihre Ovarien auszubilden [4].
Während die Königin hauptsächlich chemische Kommunikation nutzt, kommen unter Arbeiterinnen auch Tänze zum Einsatz. Diese werden primär zur Vermittlung von Futterquellen verwendet. Hierbei läuft die Entdeckerin eine Acht ab. Die Orientierung des mittleren Durchlaufs gibt Aufschluss darüber, in welche Richtung die Quelle liegt, wohingegen in der Dauer des Durchlaufs die Entfernung kodiert ist. Beides ist in Abbildung 1 (Seite 2) zu erkennen. Als Reaktion auf einen solchen Tanz beginnen einige Arbeiterinnen der Tänzerin zu folgen. Dabei nehmen sie Aromen auf, welche die Tänzerin aufgenommen haben und dabei helfen, die Quelle genauer zu lokalisieren.

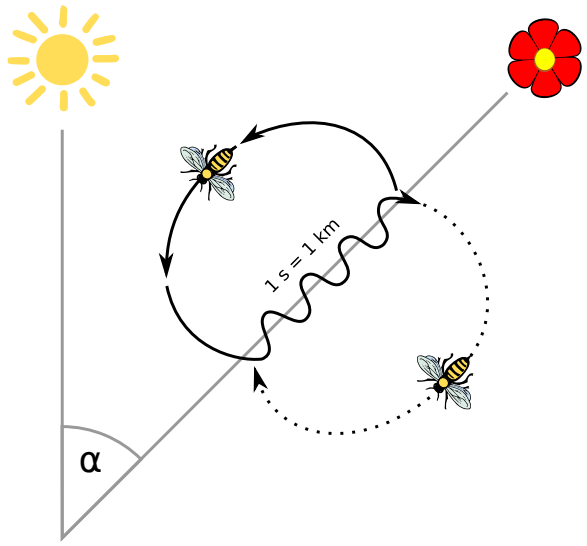

Abbildung 1: Schematische Darstellung des Schwänzeltanzes [5] – die Orientierung der Acht gibt die Richtung der Nahrungsquelle als Winkel zur Sonne an.

Auch wenn die Kommunikation von Bienenvölkern Subjekt zahlreicher Forschungsarbeiten war, gibt es immer noch einige Fragen die es zu klären gibt. Eine davon lautet: welchen Einfluss haben Beziehungen zwischen den Mitgliedern eines Stocks auf die Kommunikation? Oder auch einfacher: bilden Bienen Cliquen?

2 Das Projekt

Der Vorläufer von Beesbook[1] war Robobee[2]. Ziel von Robobee war die Imitation des Schwänzeltanzes, um neue Erkenntnisse über diesen Tanz zu sammeln. Im Rahmen der Robobee ist es gelungen eine künstliche Biene zu entwickeln, dem einige Arbeiterinnen uneingeschränkt folgen [6]. Jedoch hat sich gezeigt, dass es komplizierter ist Arbeiterinnen dazu zu animieren die Verfolgung aufzunehmen.
Das Beesbook-Projekts versucht soziale Netzwerke zwischen Bienen zu finden und zu klären, welchen Einfluss diese auf ihr Kommunikationsverhalten haben.

2.1 Versuchsaufbau

Um das Innenleben eines Bienenstocks zu beobachten wird ein sog. *Schaukasten* genutzt. Dieser besteht aus einem Holzrahmen in den zwei Waben übereinander eingehängt sind. Glasscheiben bilden die Grenze zur Außenwelt. In einem solchen Schaukasten kann das Bienenvolk eine Größe von ungefähr 2000 Mitgliedern erreichen.

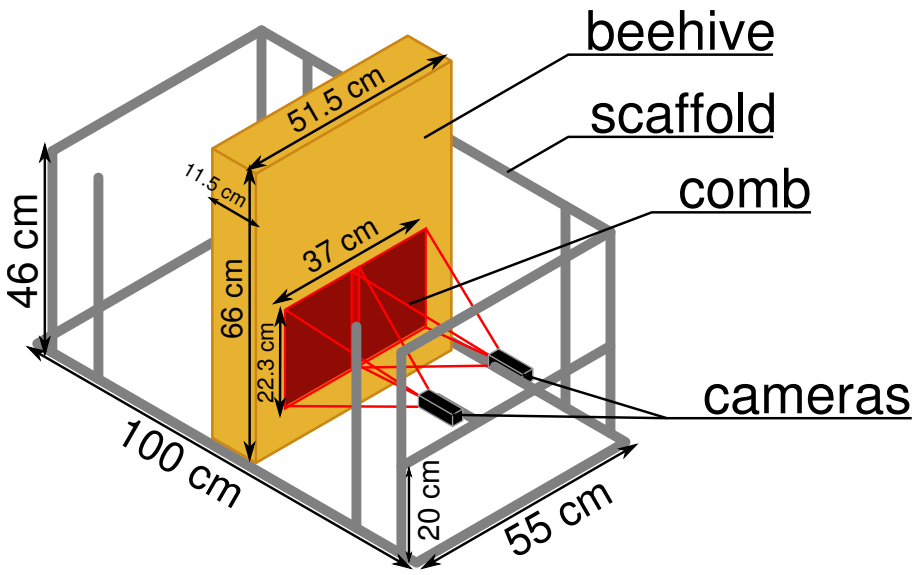

Abbildung 2: Der Bienenstock ist in einem Aluminiumrahmen befestigt. Auf die untere Wabe sind vier Kameras gerichtet – zwei auf jeder Seite, welche Bilder im Infrarot-Bereich aufnehmen.

[1] http://robobiene.mi.fu-berlin.de/wordpress/
[2] http://beesbook.mi.fu-berlin.de/wordpress/

In Abbildung 2 (Seite 3) ist der Versuchsaufbau dargestellt. Es wurde die untere Wabe gewählt, da es sich dabei um eine Brutwabe handelt, auf der sich ein Großteil des Volkes aufhält. Eine Beobachtung der oberen Honigwabe würde eine doppelt so hohe Datenmenge bedeuten, wobei der Nutzen durch die niedrige Bienendichte relativ gering wäre.
Jede Kamera hat eine Auflösung von 12 Megapixel und deckt ungefähr die halbe Wabe ab. Die Bilder werden im Infrarot-Bereich aufgenommen, da Bienen Licht in diesem Spektrum nicht wahrnehmen können [7]. Jede Kamera führt vier Aufnahmen pro Sekunde durch.

2.2 Tag Design

Einzelne Individuen eines Volkes können von Menschen, zumindest ohne Hilfsmittel, nicht unterschieden werden. Ebenso ist uns auch keine automatisierte Methode bekannt, einzelne Bienen zu identifizieren. Aus diesem Grund müssen alle Mitglieder des Volkes – oder zumindest ein Großteil davon – markiert werden. Hierzu wurde das in Abbildung 3 (Seite 4) abgebildete Tag Design konzipiert.

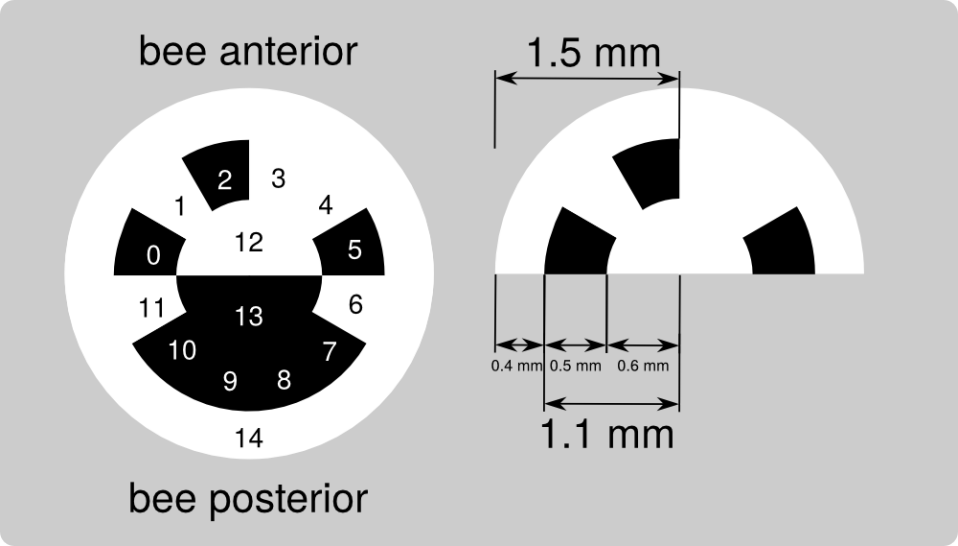

Abbildung 3: Das Tag Design mit dem die Bienen markiert werden. Jedes Tag besteht aus 12 Zellen, wobei eine weiße Zelle eine 1 repräsentiert und schwarz gleichbedeutend mit einer 0 ist. Die Zelle mit der Beschriftung 0 ist dabei das Bit mit der höchsten Wertigkeit (MSB).

Jedes Tag repräsentiert eine Zahl im Intervall $\left[0,\ 2^{12}-1\right]$. Die mittleren Halbkreise dienen, neben der Festlegung des Most Significant Bits (*MSB*), auch der Ermittlung der Orientierung der Biene.

Die Form des Tags ist der Form des Bienenthorax geschuldet, auf dem die Tags letztendlich befestigt werden. Ein Tag sollte zusätzlich eine Wölbung haben, um so gut wie möglich am Bienenthorax haften zu können. Die dadurch auftretende Deformation des Aufdrucks ist bei einem kreisförmigen Tag Design weniger kritisch.

2.3 Dekodierungsprogramm

Die Dekodierungssoftware ist, wie in Abbildung 4 (Seite 6) zu erkennen ist, nach dem *Pipes-and-Filter* Prinzip aufgebaut. Die Implementierung ist in C++, unter Zuhilfenahme des OpenCV Framework[3], geschrieben. Dieser Aufbau ist eine Weiterentwicklung der in [8] beschriebenen Software.

Ausgangspunkt ist ein Graustufenbild, in dem zu aller erst Regionen ausgemacht werden, welche ein Tag enthalten könnten – sog. *Region of Interest*, kurz *ROI*. Innerhalb dieser ROIs müssen wiederum die eigentlichen Tags gefunden werden. Da sich der weiße Rand eines Tags abheben sollte, wird dies durch eine Ellipsenerkennung durchgeführt. Eine Ellipsenerkennung ist notwendig, da die Kreisform durch die verschiedenen Positionierungsmöglichkeiten der Bienen relativ selten erhalten bleibt. Anschließend wird das Bild so transformiert, dass das Tag wieder eine Kreisform erhält. Als Nächstes werden im *Grid Fitting* im transformierten Bild die Orientierung des Tags, sowie die einzelnen Zellen identifiziert. Diese Information wird dann im *Tag Decoding* genutzt, um die eigentliche Zahl hinter dem Tag zu ermitteln. Letztendlich werden dann die Dekodierungen auf Unstimmigkeiten – z.B.: gleiche Dekodierungen an mehreren Orten gleichzeitig – überprüft und von diesen bereinigt, sodass die Ergebnisse dann gespeichert werden können.
Diese Arbeit beschäftigt sich mit der zeitlichen und der qualitativen Optimierung der Schritte *Orientation & Grid Fitting* und *Tag Decoding*. Aus diesem Grund werden diese Schritte hier noch einmal näher beleuchtet.

[3] http://opencv.org/

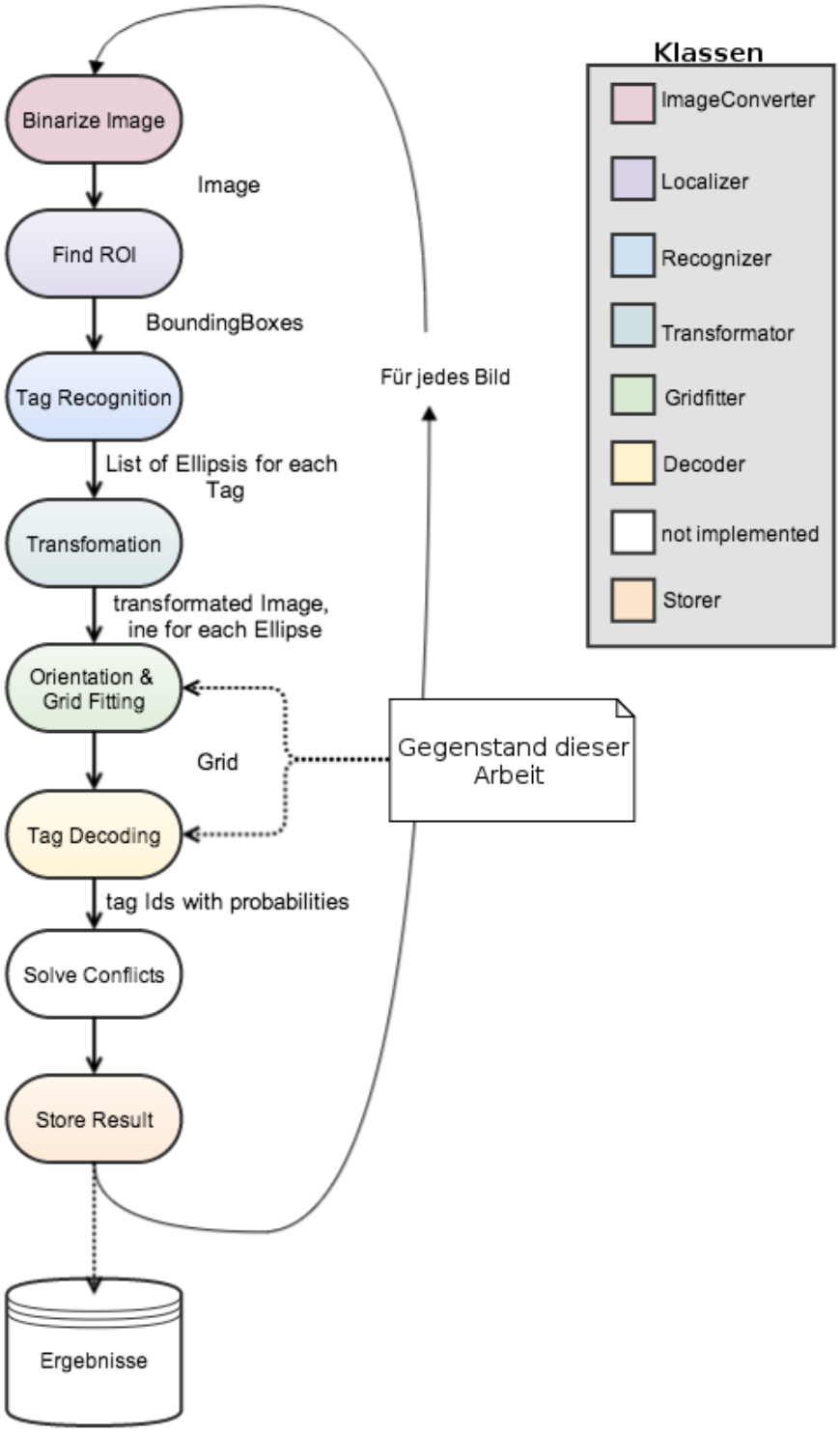

Abbildung 4: Architektur der Dekodierungssoftware – modifiziertes Diagramm nach [9]

2.3.1 Grid Fitting

Aufgabe dieses Schritts ist das *Aufsetzen* eines Gitters (*Grid*) in der Form des Tag Design, um die einzelnen Zellen identifizieren zu können. Hierzu wird dem *Grid Fitter* ein transformiertes Bild übergeben.
In diesem Schritt müssen im Grunde zwei Probleme gelöst werden:

- Lokalisierung des Mittelpunkts, da durch die Transformation der Mittelpunkt des Tags nicht unbedingt der Mittelpunkt des Bildes ist
- Ermitteln der Orientierung

Eine erste grobe Orientierung wird bereits mit Hilfe des Moments im binarisierten Bild gefunden. Anschließend wird nach dem Maximum einer Bewertungsfunktion (*Scoringunktion* oder *Scoring*) gesucht, welche von der Position des Mittelpunkts und der Rotation des Tags abhängt. Das Scoring wird im folgenden Absatz in groben Zügen beschrieben.

Sei μ_i der Mittelwert der Zelle i, C_w die Menge der als weiß bzw. C_b die Menge der als schwarz klassifizierten Zellen. Seien μ_w und μ_b die Mittelwerte der Klassen C_w und C_B. Die Mittelwerte der Zellen μ_i werden, mit Hilfe von *k-means*, in die zwei Klassen geclustert, wodurch die einzelnen Zellen nun entweder Teil von C_w oder C_b sind. Anschließend wird der Mittelwert für C_w berechnet:

$$\mu_w = \frac{1}{|C_w|} \cdot \sum_{\mu_j \in C_w} \mu_j \tag{1}$$

Daraufhin wird die Varianz σ_w^2 ermittelt:

$$\sigma_w^2 = \frac{1}{|C_w|} \cdot \sum_{\mu_j \in C_w} (\mu_j - \mu_w)^2 \tag{2}$$

Mit C_b wird analog verfahren.

Der Score als solches ist letztendlich das *Fisher-Kriterium* [10]:

$$S = \beta \cdot \frac{|\mu_w - \mu_b|^2}{\sigma_w^2 + \sigma_b^2} \tag{3}$$

wobei β ausschließlich eine lineare Skalierung ist, um evtl. zusätzliche Informationen in den Score aufnehmen zu können.

Das Maximum für S wird in einem Brute-Force-Verfahren ermittelt. Dabei wird ein kreisförmiger Bereich in der Mitte des Bildes Pixel für Pixel abgelaufen, wobei für jedes Pixel wiederum 30 verschiedene Grids, je 1° zueinander rotiert, untersucht werden. Das Grid, welches das höchste S vorweisen kann, wird für den anschließenden Schritt verwendet.

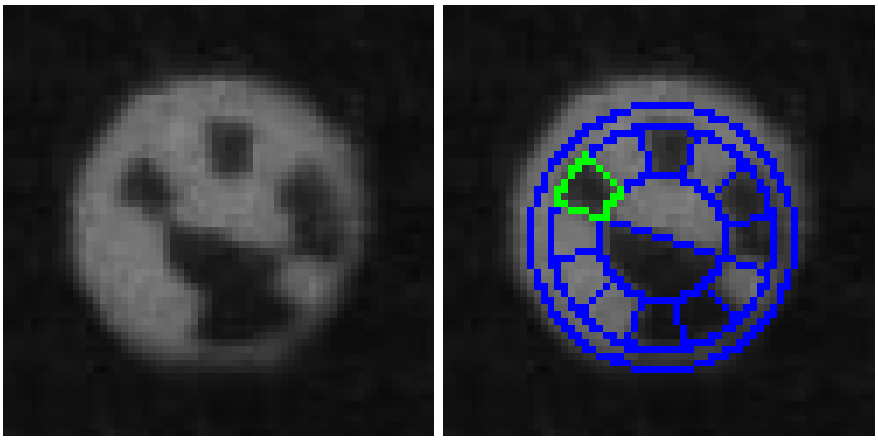

Abbildung 5: Transformiertes Bild (links) und transformiertes Bild mit dazugehöriges Grid (rechts). Die die blauen Linien sind die Umrandungen des einzelnen Zellen. Die Zelle mit dem grünen Rahmen ist dabei das MSB.

2.3.2 Tag Decoding

Dieser Schritt ist für die eigentliche Dekodierung der Tags verantwortlich und wird auch *Bitcalling* genannt. Hierzu erhält der *Tag Decoder* das transformierte Bild, sowie das Grid aus dem vorherigen Schritt. Es werden wieder die Mittelwerte der einzelnen Zellen berechnet, jedoch wird diesmal kein Clustering genutzt, um ihre Farbe zu bestimmen. Stattdessen wird ein statischer Grenzwert (*Threshold*) verwendet: sollte der Mittelwert einer Zelle i über diesem Threshold liegen, wird diese Zelle als weiß klassifiziert. Sollte der Mittelwert drunter liegen, wird die Zelle als schwarz erkannt.

3 Optimierungen

So gut die alten Implementierungen des Grid Fitting und des Tag Decoding funktioniert haben, so gab es auch einige Nachteile. So ließ die Genauigkeit der Dekodierungen zu wünschen übrig und auch die Laufzeit war deutlich zu hoch. Des Weiteren mussten diese beide Schritte an die neue Softwarearchitektur angepasst werden. So sollte z.B. das Grid Fitting mehrere mögliche Ellipsen für ein Tag als Eingabe erwarten und mehrere mögliche Grids zurückgeben. Analog dazu soll sich auch das Tag Decoding verhalten, um einen Informationsverlust durch diese Verarbeitungsschritte so gering wie möglich zu halten.

3.1 Vorbereitungen

Da eine Implementierung der in Abbildung 4 (Seite 6) gezeigten Architektur zur Entstehungszeit dieser Arbeit noch nicht vorhanden war, waren einige Vorbereitungen notwendig.

Um eine Grundlage für zukünftige Tests zu schaffen, habe ich mich dafür entschieden erst einmal die, durch das ursprüngliche Programm generierten, Ellipsen mit Hilfe der Boost-Bibliothek[4] zu serialisieren. Dadurch war ich nicht mehr auf die restlichen Teile der alten Software angewiesen und konnte mich auf die wesentlichen Schritte konzentrieren.

Als nächstes habe ich einen rudimentären Testtreiber programmiert, welcher mir die serialisierten Ellipsen wieder deserialisiert und diese dem Grid Fitter übergibt. Zusätzlich übernahm er vorübergehend die Koordination zwischen Grid Fitter und Decoder und bot mir eine Plattform zum generieren von Daten für die Evaluation.

3.2 Grid Fitting

Die Genauigkeit des Grit Fittings war bereits recht hoch, jedoch wurde ein Großteil der Laufzeit allein durch diesen Schritt in Anspruch genommen. Dies lag vor allem am Brute-Force-Ansatz, sowie an der aufwendigen Scoringfunktion. Aus diesem Grund sollten beide Teilschritte hauptsächlich beschleunigt werden, ohne dabei all zu große Einbußen in der Genauigkeit hinnehmen zu müssen.

[4] http://www.boost.org/doc/libs/1_37_0/libs/serialization/doc/index.html

3.2.1 Gradientenabstieg

Die erste Maßnahme war die Implementierung eines Gradientenabstiegs bzw. -aufstiegsverfahren [11]. Ich verwende im weiteren Verlauf ausschließlich den Begriff Gradientenabstieg.

Das Tag wird gewissermaßen in einem 4-Dimensionalen Raum betrachtet, dessen Achsen die x-Ordinate im Bild, y-Ordinate im Bild, der Winkel a des Grids und der Score S dieses Grids sind. Da der Score, zumindest indirekt, von den anderen Größen abhängt, kann man ihn auch als Funktion $S(x,\, y,\, a)$ betrachten. Da uns dieser Zusammenhang jedoch nicht direkt bekannt ist, müssen die Parameter von $S(x,\, y,\, a)$ Schrittweise modifiziert werden. Dadurch kann man sich Schrittweise einem Maximum nähern.

Eine erste Version des Gradientenabstiegs G_1 hat sequenziell jeden Parameter einzeln um eine Einheit geändert. So wird zuerst das Maximum von $\{S(x-1,\, y,\, a),\, S(x,\, y,\, a),\, S(x+1,\, y,\, a)\}$ ermittelt und der x-Parameter übernommen, der zum besten Score gehört. Anschließend wird analog mit y und a verfahren. Dies wird solange wiederholt, bis keine Änderungen mehr an der Parametern stattfinden und somit ein Maximum gefunden wurde.

Eine zweite Version G_2 orientiert sich an der RPROP-Variante des Backpropagation-Algorithmus für neuronale Netze [12]. Die Vorgehensweise wird im Folgenden in Pseudocode dargestellt, wobei $i \in \mathbb{N} : a_i$ hierbei den Winkel beschreibt, bei dem der Score bei gegebener Position sein Maximum hat:

1. gegeben ist eine initiale Startposition x, y, sowie eine Schrittweite ϵ, eine Abbruchschrittweite ϵ_0, eine Beschleunigung $\eta_{up} > 1$ und eine Bremse $0 < \eta_{down} < 1$
2. es werden folgende Scores berechnet:
   $M = \{S(x-\epsilon,\, y,\, a_1),\, S(x,\, y+\epsilon,\, a_2),\, S(x+\epsilon,\, y,\, a_3),\, S(x,\, y-\epsilon,\, a_3)\}$
3. sollte $\max(M)$ besser als $S(x,\, y,\, a_0)$ sein:
   (a) setze x und y auf die Position von $\max(M)$
   (b) setze $\epsilon := \epsilon \cdot \eta_{up}$
4. sonst:
   (a) behalte momentane Position bei
   (b) setze $\epsilon := \epsilon \cdot \eta_{down}$
5. sollte $\epsilon > \epsilon_0$ sein: springe zurück zu Schritt 2

Aus Übersichtsgründen habe ich an dieser Stelle darauf verzichtet, die Ermittlung des optimalen Winkels a_i zu beschreiben. Hier kommt ein eindimensionaler Gradientenabstieg zum Zug, wobei das Vorgehen analog zum gerade beschriebenen Algorithmus ist.
Mit der Nutzung variabler Schrittweiten sollen lokale Maxima übersprungen werden. Nichtsdestotrotz entscheidet im wesentlichen der Startpunkt über den Erfolg des Gradientenabstieg. Um diesem Problem zu begegnen, wird der Algorithmus mehrmals mit zufällig generierten Startpositionen durchgeführt, in der Hoffnung zumindest einen vorteilhaften Start zu haben.

3.2.2 Scoring

Ein weiterer Ansatz zum Optimieren des Grid Fittings waren Änderungen am Scoring. Zum Einen sollte der Genauigkeitsverlust durch den Gradientenabstieg zu kompensiert werden – laut [8] betrug die ursprüngliche Fittingquote 93,94 %.
Zum Anderen hat das Scoring den Großteil an der Laufzeit ausgemacht. So hat die Untersuchung des Laufzeitprofil mit *gprof*[5] ergeben, dass knapp 95 % der Zeit ins Scoring geflossen ist.

Das neue Scoringverfahren stützt sich auf bereits binarisierte Bilder. Hierzu kam ursprünglich die Otsu-Binarisierung [13] zum Einsatz. Dieses wurde jedoch von einem lokalen Verfahren verdrängt, welches mit den verwendeten Algorithmen bessere Ergebnisse lieferte. Hierbei hat sich ein Fenster mit einer Größe von 21 Pixel am günstigsten erwiesen, wobei als Schwellwert ein gewichteter Mittelwert der Werte innerhalb des Fensters genutzt wird. Die Gewichtung findet dabei über eine Normalverteilung statt.

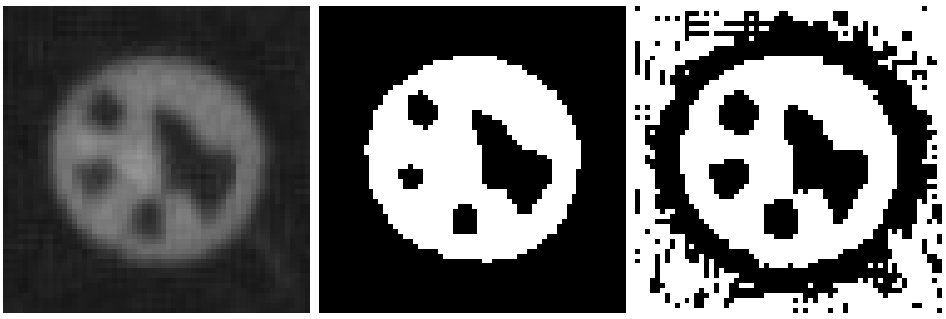

Abbildung 6: Das linke Bild ist das ursprüngliche Graustufenbild, in der Mitte das mit Otsu binarisierte Bild und Rechts das Binärbild unter Verwendung des lokalen Verfahrens.

Das binarisierte Bild wird genutzt, um die Anzahl der weißen und schwarzen Pixel in einer Zelle zu zählen. Anschließend wird ein Quotient aus beiden

[5] http://sourceware.org/binutils/docs/gprof/

Zahlen gebildet, wobei die Kleinere der Dividend ist. Die Summe alles Quotienten bildet dann den Score S_1:

$$S_1 = \sum_{i=0}^{11} \frac{\min(c_{si}, c_{wi})}{\max(c_{si}, c_{wi})} + \sum_{i=12}^{13} \frac{c_{si}}{c_{wi}} + \frac{c_{w14}}{c_{s14}} \tag{4}$$

wobei c_{si} die Anzahl der schwarzen und c_{wi} die Anzahl der weißen Pixel in Zelle i sind. Eine besondere Rolle wird dabei den Zellen 12, 13 und 14 zuteil, da deren Farbe bereits durch das Tagdesign festgelegt ist. Diese sollten also von sich aus eine größere Anzahl weißer oder schwarzer Pixel beinhalten.

Eine alternative Variante S_2 ignoriert die Zellen im mittleren Ring. Es wird also nur der äußere weiße Ring, sowie der innere weiße Halbkreis und schwarze Halbkreis betrachtet:

$$S_2 = \sum_{i=12}^{13} \frac{c_{si}}{c_{wi}} + \frac{c_{w14}}{c_{s14}} \tag{5}$$

Beide Scores konvergieren im Idealfall gegen 0.

Die Idee hinter beiden Varianten ist es eine relativ einfache und schnelle Metrik für die "Reinheit"der Zellen bereit zu stellen. Dabei soll der weiße Ring die korrekte Positionierung gewährleisten, während die Halbkreise für die richtige Orientierung verantwortlich sein sollen.

3.3 Tag Decoding

Im Gegensatz zum Grid Fitting, gab es beim Tag Decoding kein Potential zur Beschleunigung. Jedoch ließ die Genauigkeit mit 53,68 % zu wünschen übrig. Deswegen mussten Verfahren implementiert werden, welche robuster als die Nutzung eines statischen Threshold sind.

3.3.1 Halbkreise als Referenzwerte

Ein erster Versuch war das Nutzen der mittleren Halbkreise als Referenzwerte für die einzelnen Zellen. Hierzu wurde zuerst der Mittelwert über den Bereich jeweils eines Halbkreises berechnet. Anschließend wurden die Mittelwerte für die einzelnen Zellen ermittelt und die Zellen, entsprechend dem euklidischen Abstands zu den Mittelwerten der Halbkreise, klassifiziert. Dabei wurde eine Zelle als schwarz klassifiziert, wenn der Mittelwert dem Mittelwert des schwarzen Halbkreis näher lag. Analog dazu verlief die Klassifizierung als weiß.
Für die Berechnung der Mittelwerte wurde hierbei ein verkleinerter Bereich in der Mitte der Zellen betrachten, um den Einfluss angrenzender Zellen auf die Klassifizierung möglichst klein zu halten.

3.3.2 Include-Exclude-Methode

Eine Weiterentwicklung des vorangegangen Verfahrens ist die *Include-Exclude-Methode*. Hierzu werden vorläufig wieder alle Zellen mit Hilfe der Referenzwerte der inneren Halbkreise klassifiziert. Anschließend beginnt das verschieben einzelner Zellen von einer Klasse in die andere. Zu aller erst wird hierbei wieder ein Fisher-Score F_0, ähnlich wie in (3), berechnet. Hierbei werden jedoch die Mittelwerte und Varianzen über alle Pixel einer Klasse berechnet. Die Klasse eines Pixel entspricht dabei der Klasse der Zelle, in dessen Fläche das Pixel liegt. Danach wird die schwarze Zelle mit dem höchsten Mittelwert (die hellste schwarze Zelle) in die Klasse der weißen Zellen verschoben und der Fisher-Score $F_{b\to w}$ der neuen Klassifizierung berechnet. Die Zelle wird also aus ihrer alten Klasse entfernt (*exclude*) und die neue Klasse eingefügt (*include*). Das Gleiche geschieht mit der dunkelsten weißen Zelle (der Score hierfür nenne ich $F_{w\to b}$). Am Ende wird die Klassifizierung mit dem Score $\max(F_0, F_{b\to w}, F_{w\to b})$ übernommen. Dies wird solange durchgeführt, bis $\max(F_0, F_{b\to w}, F_{w\to b}) = F_0$ gilt, also sich die momentane Klassifizierung als vorteilhafter herausstellt.

Ein Problem des Verfahrens aus Abschnitt 3.3.1 (Seite 12) war, dass weiße Zellen aus unterschiedlichsten Gründen als schwarz klassifiziert werden. Dies soll durch dieses Verfahren minimiert werden.

3.3.3 Edge-Walker

Diese Methode verfolgt einen grundsätzlich anderen Ansatz als die anderen beiden Verfahren. Anstatt Flächen zu betrachten, wird eine Kreiskante, mit Hilfe der Bresenham-Algorithmus [14], durch die Zelle gezogen. Ein Beispiel einer solchen Kreiskante ist in Abbildung 7 (Seite 14) zu sehen ist.
Beim Ablaufen dieser Kante wird die Intensität der Pixel in einer linearen Datenstruktur gespeichert. Sollte die Dicke der Kante größer als ein Pixel sein, werden die innereren Kanten auf die äußerste Kante projiziert und der Mittelwert der Intensitäten über diese Projektion verwendet. Das Ergebnis dieses Vorgehens ist ein Helligkeitsprofil entlang dieser Kante. Zwei Illustrationen eines solchen Profils sind in Abbildung 8 (Seite 15) dargestellt.

Als Ersten Schritt erfolgt die Ermittlung eines Schwellwertes c, der im späteren Verlauf als Orientierungshilfe dient:

$$c = \frac{\max(P) + \min(P)}{2} \tag{6}$$

P ist hierbei die Menge der Intensitäten des Profils.

Innerhalb dieses Profiles werden als nächstes Extrempunkte bestimmt. Als Extrempunkt werden dabei Stellen bezeichnet, deren Steigungsrichtung zu

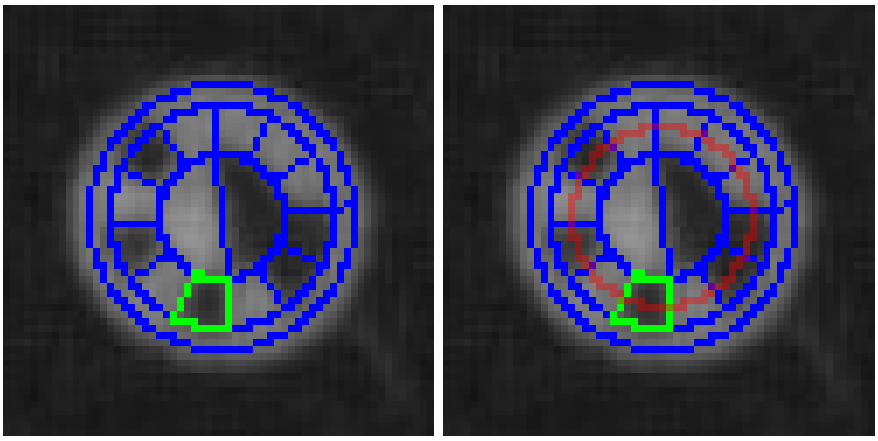

Abbildung 7: Abbildung eines Tags (links) und des gleichen Tags mit zugehöriger Kreiskante (rechts) – Kreiskante hier in leicht transparentem Rot

den Nachbarstellen gleich ist – z.B. beide Nachbarpunkte liegen oberhalb des Extrempunkts. Sollte einer der Nachbarpunkte auf der gleichen Höhe liegen, gilt für diese Seite der nächstgelegene Punkt, dessen Niveau sich unterscheidet. Das Ergebnis ist eine Liste von Extrempunkten, wobei auf ein Minimum immer ein Maximum folgt und umgekehrt.
Anschließend wird diese Liste durchlaufen. Sollte dabei c überschritten werden, wird im Bereichs des Übergang, ein *Transitionpoint* positioniert. Zusätzlich wird die Orientierung (Funktion aufsteigend bzw. absteigend) an diesem Punkt gesichert.
Im idealen Fall sollten alle Transitionpoints auf Zellgrenzen liegen. Um dies kleinere Abweichungen zu kompensieren, wird noch eine Positionskorrektur vorgenommen: die Positionen der Transitionpoints wird soweit verschoben, sodass der erste Transitionpoint auf der nächsten Zellgrenze liegt.

Jeder Transitionpoint markiert nun einen Übergang von einer Farbe zur Anderen und befindet sich im Idealfall zumindest in der Nähe einer Zellgrenze. Mit Hilfe dieses Zusammenhangs und der Orientierung der Punkte werden letztendlich die Zellen klassifiziert, wobei alle Zellen, welche sich zwischen dem gleichen Paar von Transitionpoints befinden, derselben Klasse angehören.

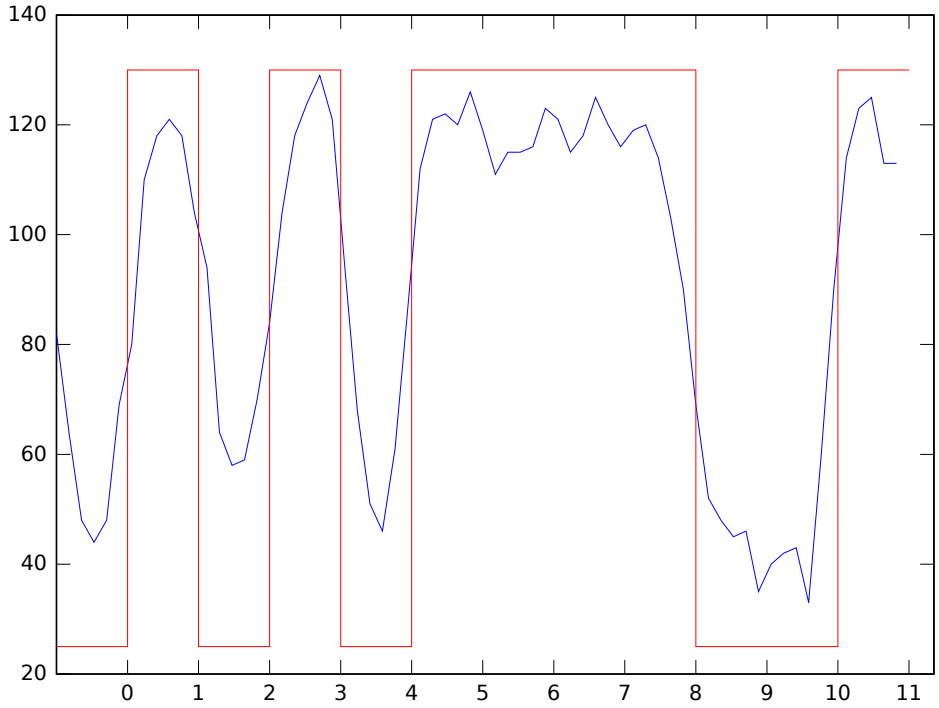

Abbildung 8: Helligkeitsprofil des Tag in Abbildung 7 (Seite 14). Der ideale Verlauf ist in Rot dargestellt. Die Beschriftung der x-Achse gibt an, welche Zelle an der entsprechenden Stelle endet.

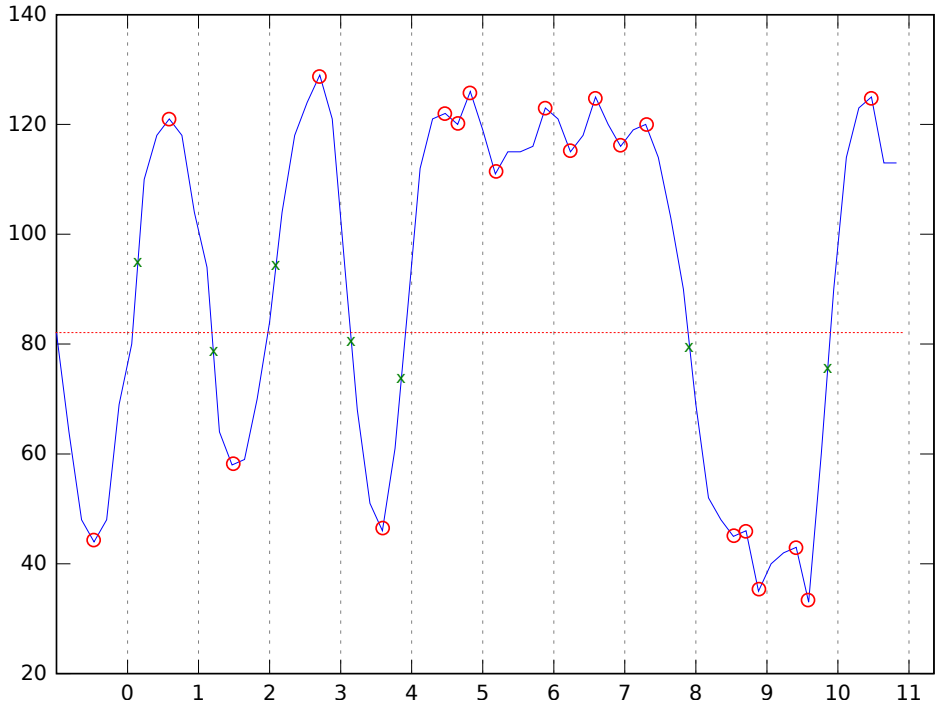

Abbildung 9: Das Ergebnis des Edge-Walker-Algorithmus im Diagramm – die rot gestrichelte Horizontale ist der Schwellwert c, die rot umkreisten Stellen sind die erkannten Extrempunkte und die grünen x die Transitionpoints.

3.4 Reduktion der Dekodierungen

Das Grid Fitting erwartet n Ellipsen und gibt für jede Ellipse drei mögliche Grids an den Decoder weiter. Der Decoder wiederum wendet sowohl die Include-Exclude-Methode, als auch den Edge-Walker auf jedes Grid an. Die Anzahl der Dekodierungen d ist also durch folgenden Zusammenhang definiert:

$$d = n \cdot 3 \cdot 2 = 6n \tag{7}$$

Durch diese Menge an möglichen Dekodierungen benötigt man ein Bewertungsmaß. Hierfür hat sich abermals der Fisher-Score aus Abschnitt 3.3.2 (Seite 13) angeboten.

4 Evaluation

Da mir zum Zeitpunkt der Anfertigung dieser Arbeit noch keine Aufnahmen eines echten Bienenvolks zur Verfügung standen, wurden einige Tags auf einem schwarzes Tuch befestigt. Durch Strecken und Stauchen wurden die Tags in verschiedenen Winkeln zur Betrachtungsebene positioniert. Davon wurden JPEG-Bilder mit dem Qualitätslevel 90 angefertigt.
Anmerkung: Die Werte in den Tabellen sind bei stochastisch ermittelten Ergebnissen mit der Standardabweichung hinter dem ± angegeben.

4.1 Grid Fitting

Bereits eine erste Sichtung der Ergebnisse von G_1 hat gezeigt, dass dieses Verfahren ungeeignet war. Es wurden nur lokale Maxima gefunden, die nur in den seltensten Fällen geeignet waren. G_2 hingegen hat von vorneherein bessere Ergebnisse geliefert, weshalb nur dieses Verfahren für den Gradientenabstieg evaluiert wurde.

4.1.1 Laufzeit

Wie in Tabelle 1 (Seite 17) zu sehen ist, ist G_2 um 91 % schneller als die Bruteforce-Methode. Auch das mehrmalige Wiederholen des Grid Fitting von anderen Startpunkte aus konnte diese Zeitersparnis nicht komplett auffressen. So benötigt selbst das 13-fache Wiederholen $(1{,}682 \pm 0{,}365)$ s, was immer noch fast doppelt so schnell ist wie das Vorgehen mit Bruteforce.

	Brute Force	Gradientenabstieg
Laufzeit in s	$3{,}260 \pm 0{,}372$	$0{,}281 \pm 0{,}073$

Tabelle 1: Durchschnittliche Laufzeit des gesamten Grid Fitting mit der alten Brute Force Methode und dem Gradientenabstieg. Gradientenabstieg ist dabei G_2 mit einer Startposition. Als Scoring Methode wird bei dieser Messung das alte Scoring genutzt.

	Altes Scoring	S_1	S_2
Laufzeit in s	$0{,}453 \pm 0{,}226$	$0{,}271 \pm 0{,}067$	$0{,}090 \pm 0{,}032$

Tabelle 2: Durchschnittliche Laufzeit des gesamten Grid Fitting für ein Tag, unter Zuhilfenahme der angegebenen Scoringfunktionen und des Gradientenabstiegverfahrens G_2. Der angegebene Fehlerbereich ist hierbei die Standardabweichung der Messung.

Auch beim Scoring konnte sehr viel Zeit eingespart werden. So benötigt S_1 40 % weniger Zeit, bei S_2 sind es sogar 80 % (Tabelle 2 (Seite 17)). Durch die zusätzliche Zeitersparnis beim Scoring konnten noch weitere Startpunkte für den Gradientenabstieg genutzt werden. Die Ergebnisse dieser Zeitmessung sind in Tabelle 3 (Seite 18) zu sehen.

	13 Starts	17 Starts	21 Starts
Laufzeit in s	$0{,}298 \pm 0{,}067$	$0{,}381 \pm 0{,}089$	$0{,}416 \pm 0{,}081$

Tabelle 3: Durchschnittliche Laufzeit des Grid Fittings unter Verwendung des Gradientenabstiegs G_2 und der Scoringfunktion S_2 mit unterschiedlicher Anzahl zufällig gewählter Startpunkte.

Die Konfiguration mit den 17 Startpunkten hat sich, im späteren Verlauf, am günstigsten erwiesen. Bezogen auf diese Zeit, konnte im Große und Ganzen konnte insgesamt 88,5 % der Laufzeit beim Grit Fittings eingespart werden.

4.1.2 Genauigkeit

An den blauen Balken in Abbildung 10 (Seite 18) ist zu erkennen, dass die Genauigkeit von über 90 % [8] deutlich abgenommen hat. Interessanterweise verträgt sich der Gradientenabstieg mit den neuen Scoringverfahren etwas besser. S_1 weist zwar die beste Trefferquote auf, jedoch sind weit weniger Grids nutzbar. Letztendlich liefert S_2, unter Nutzung von G_2, die höchste Anzahl nutzbarer Grids.

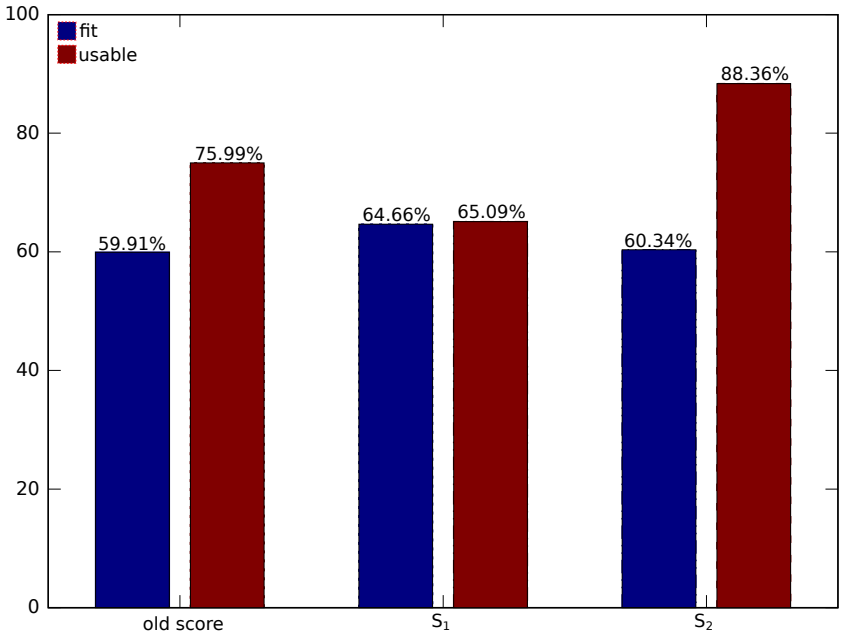

Abbildung 10: Trefferquote des Grid Fitting unter Zuhilfenahme der angegebenen Scoringfunktionen und des Gradientenabstiegverfahrens G_2 mit fünf zufälligen Startpunkten. *Fit* bedeutet, dass das Grid nahezu perfekt passt. *Usable* bedeutet, dass maximal eine Rotation von einer halben Zelle benötigt wird, um ein Fit zu erreichen.

4.2 Tag Decoding

Die Grids für das Tag Decoding wurden in diesen Tests mit G_2 unter Nutzung der Scoringfunktion S_2 generiert. Da das Grid Fitting einen randomisierten Aspekt beinhaltet, wurde das Tag Decoding mit jeder Konfiguration 20-mal wiederholt. Die Trefferquoten wurden anschließend gemittelt, um möglichst repräsentative Ergebnisse zu erhalten.

Die ursprüngliche Erkennungsrate von fast 54 % [8] wird sowohl von der Include-Exclude-Methode, als auch vom Edge-Walker um mindestens 15 % übertroffen.
Sowohl bei der Include-Exclude-Methode, als auch beim Edge-Walker scheint die Menge der eingehenden Daten kaum eine Rolle zu spielt. Eine Kombination aus beiden Verfahren liefert etwas höhere Trefferquoten (Tabelle 6 (Seite 19)). Jedoch Überschneiden sich scheinbar die Ergebnisse der beiden Verfahren weitestgehend.

	nur beste Ellipse genutzt		alle Ellipsen genutzt	
	nur beste Ellipse	Decodings	alle Ellipsen	Decodings
ein Grid	$84{,}83 \pm 0{,}02$	$1{,}00 \pm 0{,}00$	$84{,}35 \pm 0{,}02$	$1{,}27 \pm 0{,}47$
drei Grids	$84{,}68 \pm 0{,}02$	$3{,}00 \pm 0{,}00$	$84{,}96 \pm 0{,}02$	$3{,}81 \pm 1{,}42$

Tabelle 4: Trefferquote der Include-Exclude-Methode in Prozent. Die durchschnittliche Anzahl der zurückgegebenen Dekodierungen ist in den Spalten *Decodings* zu sehen. Ein Treffer liegt vor wenn mindestens eine Dekodierung korrekt ist.

	nur beste Ellipse genutzt		alle Ellipsen genutzt	
	nur beste Ellipse	Decodings	alle Ellipsen	Decodings
ein Grid	$73{,}69 \pm 0{,}02$	$1{,}00 \pm 0{,}00$	$73{,}73 \pm 0{,}02$	$1{,}27 \pm 0{,}47$
drei Grids	$73{,}56 \pm 0{,}02$	$3{,}00 \pm 0{,}00$	$72{,}78 \pm 0{,}02$	$3{,}81 \pm 1{,}42$

Tabelle 5: Trefferquote der Edge-Walker-Methode in Prozent. Ein Treffer in verhält sich analog zur Definition in Tabelle 4 (Seite 19).

	nur beste Ellipse genutzt		alle Ellipsen genutzt	
	nur beste Ellipse	Decodings	alle Ellipsen	Decodings
ein Grid	$86{,}08 \pm 0{,}01$	$2{,}00 \pm 0{,}00$	$85{,}86 \pm 0{,}02$	$2{,}54 \pm 0{,}95$
drei Grids	$85{,}95 \pm 0{,}02$	$6{,}00 \pm 0{,}00$	$86{,}34 \pm 0{,}02$	$7{,}63 \pm 2{,}84$

Tabelle 6: Trefferquote der Include-Exclude- in Kombination mit der Edge-Walker-Methode in Prozent. Ein Treffer in verhält sich analog zur Definition in Tabelle 4 (Seite 19).

4.2.1 Reduktion der Dekodierungen

Bei Nutzung aller verfügbaren Daten und Verfahren (wie in Tabelle 6 (Seite 19)) erhält man im Durchschnitt 7,63 mögliche Dekodierungen. Bei einer Begrenzung auf die drei, nach Abschnitt 3.4 (Seite 16), bestbewerteten Dekodierungen, ist in $(86{,}21 \pm 0{,}01)\,\%$ der Fälle mindestens eine Dekodierung richtig. Nutzt man nur die beste Dekodierung liegt die Trefferquote immer noch bei $(82{,}37 \pm 0{,}02)\,\%$.

4.2.2 Weitere JPEG-Qualitätslevel

Eine weitere interessante Frage war, ob auch JPEG-Bilder schlechterer Qualität zur Dekodierung geeignet sind. Hierzu habe ich die Bilder mit niedrigeren Qualitätslevel abgespeichert und diese erneut dekodieren lassen. Wie in Abbildung 11 (Seite 20) zu sehen ist, bleibt die Trefferrate bei einem Level von 85 konstant. Jedoch sinkt die Trefferquote bereits bei einem Qualitätslevel von 80 deutlich ab.

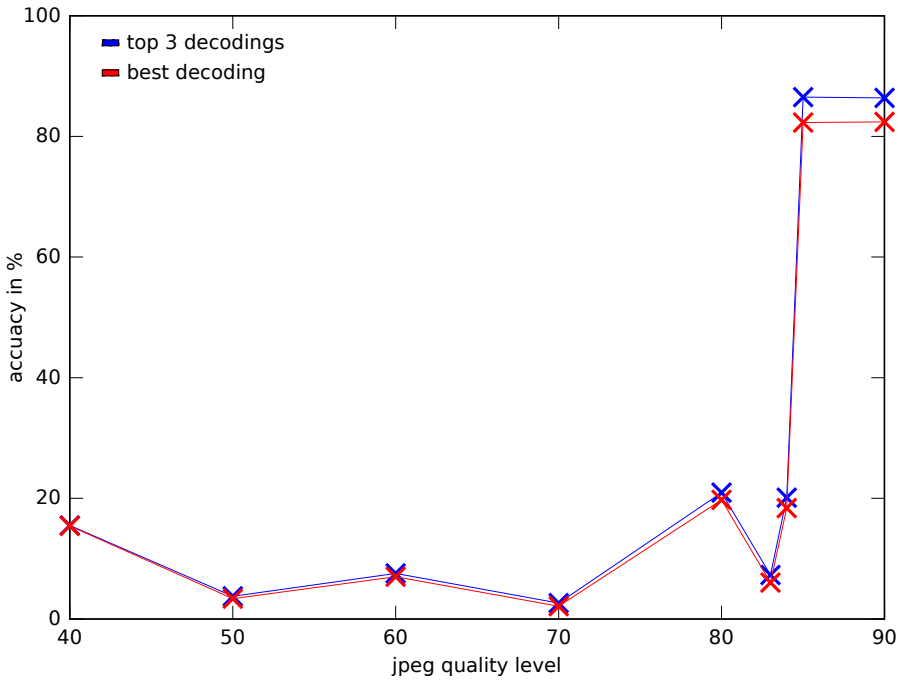

Abbildung 11: Trefferquoten in Abhängigkeit vom JPEG-Qualitätslevel

4.3 Gesamtlaufzeit

Die Gesamtlaufzeit des Grit Fitting und des Tag Decoding liegt bei $(0{,}46 \pm 0{,}17)\,\mathrm{s}$. Unter der Annahme, dass die Laufzeit des alten Tag Decoding vernachlässigbar klein ist, benötigt der gesamte Dekodiervorgang knapp 84 % weniger Zeit als vorher.

5 Fazit

Sowohl die Laufzeit, als auch die Genauigkeit konnte deutlich verbessert werden. So konnte sie Laufzeit auf weniger als ein fünftel der ursprünglichen Laufzeit reduziert werden. Auch kann die Steigerung der Genauigkeit um knapp 28 % als Erfolg gewertet werden. Nichtsdestotrotz werden noch fast 20 % der Tags nicht korrekt dekodiert. In einigen Fällen waren die Algorithmen nicht präzise genug, sei es beim Positionieren des Grids oder beim Unterscheiden von weißen und schwarzen Zellen. So war die Scoringfunktion S_2 in wenigen Fällen nicht optimal, da der Rand des Tags nicht erkennbar war.
Die meisten Fehldekodierungen waren jedoch auf kaum erkennbare Tags zurückzuführen. Dies liegt wiederum an der, kurz nach der Ellipsenerkennung durchgeführten, Transformation. Ob eine bessere Transformation möglich ist, gilt es zu diskutieren.

Desweiteren sollte der verwendete Testaufbau kritisch betrachtet werden. So besteht die Gefahr, dass ich *Overfitting* betrieben habe. Auch ist unklar wie sich viele dicht beieinander liegende Tags die Trefferquote beeinflussen können. Die Erkennungsraten könnten sich also bei Bildern aus einem echten Bienenstock noch verändern.

Die Nutzung von JPEG-Bildern niedrigerer Qualität hat sich als ungünstig erwiesen. Während ein Level von 85 noch vertretbar ist, können Bilder niedrigerer Qualität kaum noch genutzt werden.

5.1 Ausblick

Auch wenn die Laufzeitersparnis enorm ist, gibt es mit Sicherheit einige Möglichkeiten zur weiteren Optimierung. So könnten reine Arrays anstatt Matrizen oder Vektoren verwendet werden. Ebenfalls könnte man mit anderen Parametern für das Grid Fitting evtl. noch bessere Trefferquoten erreichen.

Literatur

[1] E. Crane, *The Rock Art of Honey Hunters*. Routledge Chapman & Hall, 1994.

[2] J. H. C. I. S.-D. S. A. C. C. K. Alexandra-Maria Klein, Bernard E Vaissière and T. Tscharntke, "Importance of pollinators in changing landscapes for world," *Proceedings B of the Royal Society*, 2007.

[3] M. L. W. S. A. Kolmes, *Division of labour among worker honey bees in demographically manipulated colonies*. Insectes Sociaux, 1998.

[4] M. L. W. K. N. S. Shelley E.R. Hoover, Christopher I. Keeling, "The effect of queen pheromones on worker honey bee ovary development," *Naturwissenschaften - The Science of Nature*, 2003.

[5] Kilom691, "Bee dance," 2014. [Online]. URL: https://upload.wikimedia.org/wikipedia/commons/f/f8/Bee_dance.svg

[6] Landgraf, "Robobee : A biomemetic honeybee robot for the analysis of the dance communication system," Ph.D. dissertation, Feie Universität Berlin, 2013.

[7] B. Farm, "What colors do bees see (and painting bee hives)," Juli 2012. [Online]. URL: http://brookfieldfarmhoney.wordpress.com/2012/07/21/what-colors-do-bees-see-and-painting-bee-hives/

[8] B. Laubisch, "Automatic decoding of honeybee identification tags from comb images," Februar 2014.

[9] M. Ziese, "Architektur der dekodierungssoftware," Mai 2014.

[10] M. Brown, "Fisher's linear discriminant," November 1999. [Online]. URL: http://compbio.soe.ucsc.edu/genex/genexTR2html/node12.html

[11] H. Lohninger, "Optimierungsmethoden - gradientenabstieg," August 2012. [Online]. URL: http://www.statistics4u.info/fundstat_germ/cc_optim_meth_gradient.html

[12] H. B. Martin Riedmiller, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," *IEEE Press*, 1993.

[13] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," 1979.

[14] U. Osnabrück, "Bresenham-algorithmus," Oktober 2003. [Online]. URL: http://www-lehre.inf.uos.de/~cg/2002/skript/node30.html