Diplomarbeit

Fachbereich Mathematik und Informatik
Freie Universität Berlin

Entwicklung eines visuellen Trackingsystems für biomimetische Roboterfische

Rami Akkad

Erstprüfer:	Prof. Dr. Raúl Rojas
Zweitprüfer:	Prof. Dr. Jens Krause
Betreuer:	Tim Landgraf

Berlin, März 2012

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Berlin, 12.03.2012 _____

Ort, Datum Unterschrift

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mir bei der Erstellung dieser Diplomarbeit geholfen haben und mir bei Fragen und Problemen immer geduldig beiseite standen.

Ein besonderes Dankeschön geht an:

Tim Landgraf, der mir diese Diplomarbeit ermöglicht hat, mich über die gesamte Zeit exzellent betreut hat, alle meine Fragen immer geduldig beantworten konnte und mir auch in privaten Dingen immer beiseite stand. Danke, Tim!

Hai Nguyen, der für die RoboFish 2.0 GUI und die Entwicklung des Roboters zuständig war und mit dem ich während der gesamten Zeit sehr eng zusammen gearbeitet habe. Danke für Deine Hilfe, Deine Motivation und für Deine immer währende gute Laune!

Tobias Ludwig, dessen mathematisches Fachwissen ich immer in Anspruch nehmen durfte. Danke für Deinen hilfreichen Input vor allem in der Anfangsphase und für die langjährige Freundschaft!

Michael Oertel, für die lustigen Konversationen im Büro und bei den vielen gemeinsamen Mittagessen. Vielen Dank auch an Dich für Dein physikalisches Fachwissen!

Hamid Mobalegh, für die zwei kleinen, jedoch besonders hilfreichen Hinweise bei der Belichtungszeit der Kamera und der Orientierungsfindung.

Prof. Dr. Raúl Rojas, für die Betreuung und Ermöglichung meiner Diplomarbeit und besonders für die Hilfestellung beim Paper für die Konferenz in Barcelona.

Prof. Dr. Jens Krause und *Romain Clément*, für die Idee und für die Beantwortung all meiner biologischen Fragen.

Meine Eltern, für die finanzielle und moralische Unterstützung über all die Jahre und für die Geduld.

Victoria Gotzler, für die jahrzehntelange Freundschaft und für das Korrekturlesen.

Abstract

Bei der Untersuchung von Sozialstrukturen in der Biologie können biomimetische Roboter verwendet werden. Sobald ein Roboter als konspezifisch erkannt wird, können Forscher durch dessen Interaktion mit der Tiergruppe beliebiges Verhalten studieren. In Zusammenarbeit mit Biologen der Humboldt Universität zu Berlin wird an der Freien Universität Berlin ein biomimetisches Roboterfisch-System „Robofish 2.0“, das mehrere Fisch-Replikate durch eine magnetische Kopplung an omnidirektional fahrende Roboter steuert, entwickelt. Für eine akkurate Steuerung ist ein Tracking System notwendig, das im Rahmen dieser Arbeit entwickelt wurde.

Schlagwörter: biomimetische roboter, biomimetik, schwarmintelligenz, schwarmverhalten, tracking, blobbing, binarisierung, entzerrung, rektifizierung, lokalisierung, richtungsfindung, opencv, qt, kameralinse, infrarotlicht

Inhaltsverzeichnis

1 Einleitung: Biologische Motivation und Notwendigkeit von Tracking

Um Verhaltensweisen und Interaktionen von Tieren innerhalb derer Sozialstrukturen besser verstehen zu können, wird in der Verhaltensbiologie immer öfter mit biomimetischen Robotern gearbeitet. Diese Systeme bieten Forschern im Vergleich zu konventionellen Methoden mehr Flexibilität. Sobald der Roboter als konspezifisch erkannt wird, kann der Verhaltensforscher versuchen durch direkte Interaktionen des Roboters mit der Tiergruppe spezifisches Verhalten zu untersuchen. Vielfältige und exakt reproduzierbare Interaktionsszenarien können durchprobiert werden. Es kann z.B. besser untersucht werden, welche morphologischen Eigenschaften einen Schwarmführer ausmachen. Für die genaue Reproduzierbarkeit eines Experimentes ist es in manchen Systemen unabdingbar den Roboter zu lokalisieren um ihn genau positionieren zu können. Das hier vorgestellte Trackingsystem dient als kontinuierliches Positionsfeedback für die simultane Steuerung multipler biomimetischer Roboterfische.

Im Folgenden werden das bereits vorhandene „RoboFish 1.0 "-System der Biologen und das neu entwickelte „RoboFish 2.0"-System kurz vorgestellt und es wird erklärt, weshalb ein Trackingsystem für die Steuerung des Roboters notwendig ist. Anschließend wird der Begriff Tracking erläutert und es werden einige Beispiele für den Einsatz von Tracking bei biomimetischen Robotern diskutiert. Der Versuchsaufbau für „RoboFish 2.0" wird skizziert, es wird gezeigt, wie das mit den Fischen interagierende Fisch-Replikat erstellt wird und erläutert, wie dieses vom Tracking-System lokalisiert und verfolgt wird. Anschließend wird beschrieben, wie das Tracking-System validiert wurde und die Ergebnisse werden diskutiert.

1.1 Robofish 1.0

Um zu untersuchen, welche morphologischen Eigenschaften und welches Verhalten einen Fisch als Schwarmführer auszeichnen, benutzen Biologen ferngesteuerte Nachbildungen von Fischen (Roboterfische / Fisch-Replikate, Abbildung 1), die magnetisch an einen Plotter, der sich unter dem Aquarium befindet, gekoppelt und durch diesen gesteuert werden [3].

Es wird beobachtet, ob und wie die Fische mit dem Roboterfisch interagieren. Folgen z.B. Fische dem sich bewegenden Roboterfisch, so wird dieses Szenario als erfolgreiche Interaktion gesehen. Die Forscher konnten mit dieser Methode validieren, dass die Fische die Nachbildung als konspezifisch anerkennen und mit dieser interagieren (Abbildung 1).

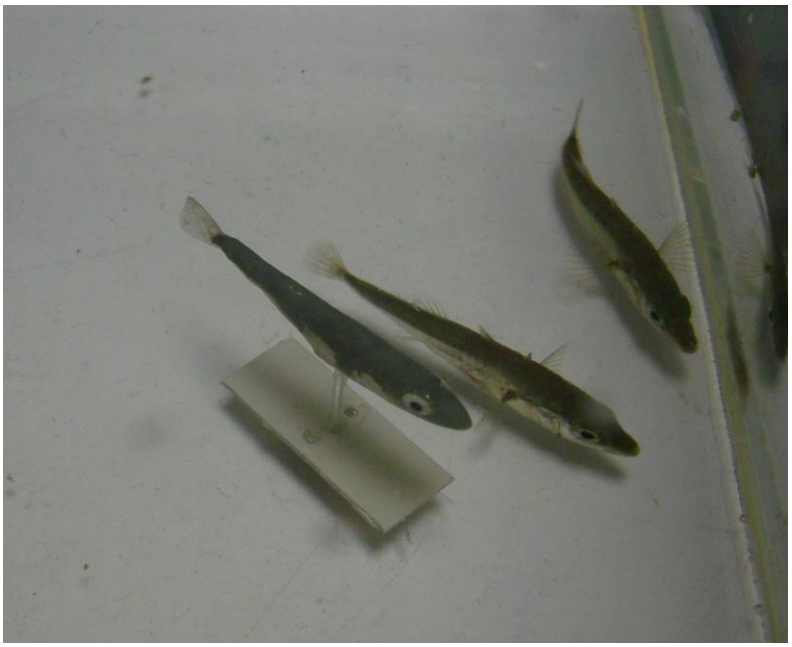

Abbildung 1 (Romain Clément): Das verwendete Replikat im Wasser mit lebenden Fischen.

1.2 Robofish 2.0

Das in Abschnitt 1.1 beschriebene „Robofish 1.0“-System hat einige Nachteile: Es kann pro Experiment maximal ein Replikat benutzt werden, da nur ein Plotter für die Steuerung des Roboterfisches zur Verfügung steht. Um zu untersuchen, welche morphologischen Eigenschaften oder welches Verhalten einen besseren Schwarmleiter ausmachen ist der simultane Einsatz mehrerer Fisch-Replikate von Vorteil, da ein direkter Vergleich zwischen der Interaktion von zwei äußerlich verschiedenen Nachbildungen mit lebenden Fischen einfacher möglich ist. Die Steuerung durch den Plotter führt aufgrund der magnetischen Kopplung zu unnatürlichen Bewegungen bei einer 180° Richtungsänderung. Diese Bewegungen wirken unter Umständen sehr aggressiv auf die lebenden Fische und stören deren natürliche Interaktion mit dem Fisch-Replikat. Da sich der Plotter auf Schienen bewegt, wird dadurch die Schwimmbahn der Nachbildung eingeschränkt. Der Roboterfisch kann beispielweise keine Kreise schwimmen.

Daher ist ein neues System notwendig, das in Zusammenarbeit mit den Biologen erarbeitet wird. Diese Arbeit beschäftigt sich mit dem Teilsystem der fortlaufenden Lokalisierung und Richtungsfindung, dem Tracking der Fisch-Replikate.

1.2.1 Notwendigkeit von Tracking

Das neue System verwendet zur Steuerung der Roboterfische omnidirektional fahrende Roboter mit drei Rädern. Ein Roboter (Durchmesser: 17 cm) soll einen oder mehrere Replikate steuern. Dabei werden die Replikate (ähnlich wie im System der Biologen) mit Magneten an den Roboter gekoppelt. Um Fahrfehler durch z.B. schwache Motoren

oder Radschlupf ausgleichen zu können, ist ein Feedback-System notwendig, das entweder die tatsächliche Drehgeschwindigkeit der Motoren und/oder die tatsächliche Position des Roboters zurückgibt. Soll der Roboter z.B. für eine bestimmte Zeit mit höchster Geschwindigkeit in eine Richtung fahren, dann summieren sich eventuell kleine Abweichungen zu einer sehr großen Abweichung (Abbildung 2, Abbildung 3) und der Roboter fährt in eine andere unerwünschte Richtung.

Da die Motoren des verwendeten Roboters kein Feedback zur aktuellen Drehzahl liefern und auch mit einem solchen Feedback die genaue Positionsbestimmung (z.B. durch sich summierende Fehler bei der Korrektur der Drehzahlen) zu ungenau werden kann, ist eine fortlaufende Lokalisierung und Orientierungsfindung, also ein Tracking des Roboters, unabdinglich. Die Position und die Richtung werden im Tracking konstant aktualisiert und können mit Hilfe eines Reglers zur Positionskorrektur verwendet werden.

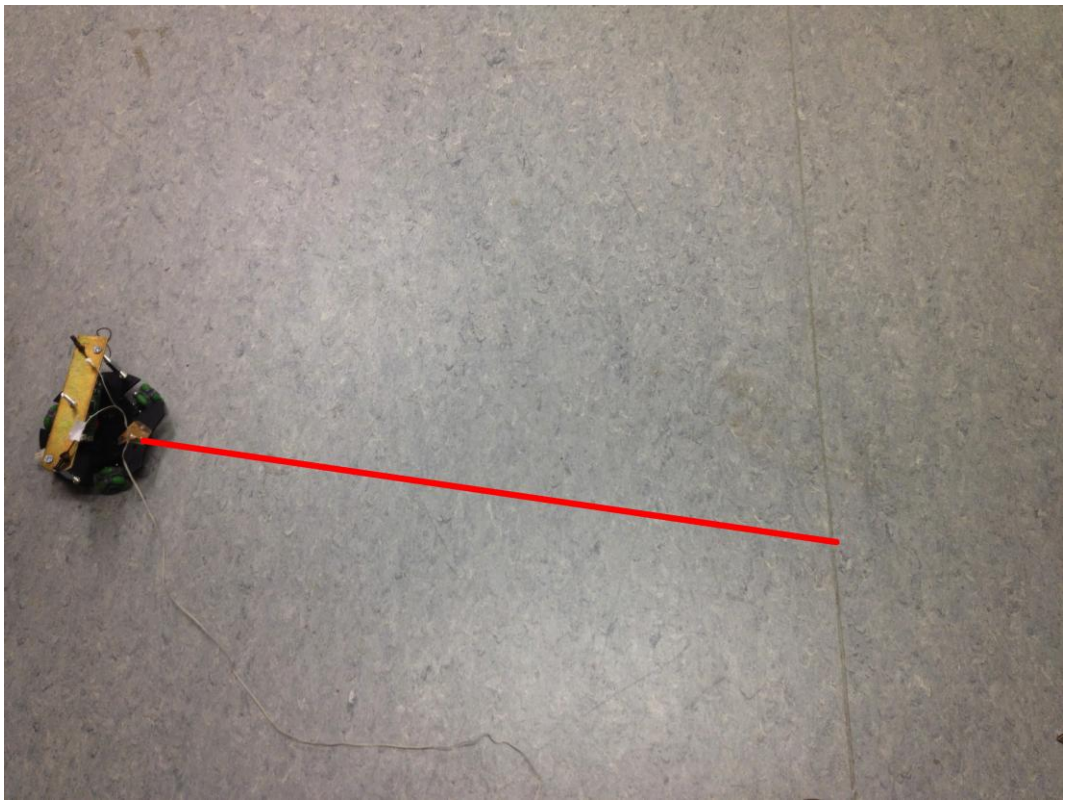

Abbildung 2: Links ist der Roboter zu erkennen (Durchmesser: 170 mm). Die rote Linie stellt die Trajektorie dar, die abgefahren werden soll.

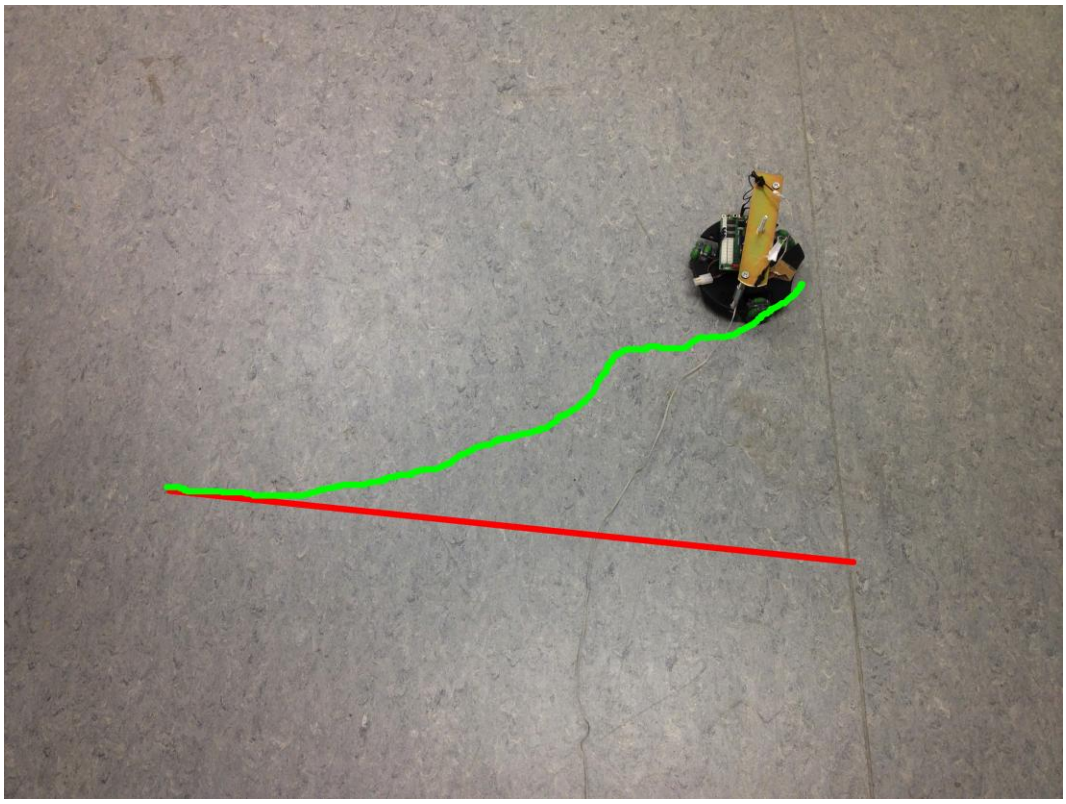

Abbildung 3: Der Roboter (Durchmesser: 17 cm) nach der abgefahrenen Trajektorie (grüne Linie) aus Abbildung 2 (rote Linie). Eine große Abweichung ist sichtbar (>34 cm).

1.3 Tracking

Als Tracking wird die fortlaufende Lokalisierung von Objekten in einer Reihenfolge von Bildern bezeichnet [2]. Dabei muss jedes einzelne Bild zuerst vorbearbeitet (optimiert) werden, um dann die zu lokalisierenden Objekte herauskristallisieren zu können (siehe Abschnitte 2.3 bis 3.3). Wurden in jedem Bild alle Objekte gefunden, beginnt die eigentliche Aufgabe des Trackings, nämlich die Zuordnung von identischen Objekten in der Bildfolge [2]. Dazu wird die Bewegung eines jeden Objektes zwischen einer Abfolge von Bildern festgestellt und es werden Ausnahmebehandlungen bei Verlust der Lokalisierung getroffen (siehe Abschnitt 3.4.3), die in der Regel mit einer Annahme bezüglich des verlorenen Objektes (z.B. geschätzte Position, erwartete Orientierung) arbeiten [2].

1.4 Beispiele für den Einsatz von Tracking bei biomimetischen Robotern

Biomimetische Roboter werden in der Wissenschaft in unterschiedlichen biologischen Sozialsystemen eingesetzt. An der Freien Universität wurde in Zusammenarbeit mit Biologen eine Roboterbiene entwickelt, die den Bienentanz nachahmen kann (Abbildung 4). Die robotische Biene hat die Aufgabe anderen Bienen die Position einer

Futterquelle zu berichten. Anhand künstlicher Stimulus-Kombinationen, die in natürlichen Tänzen so nicht vorkommen, wird versucht, die essentiellen Signale des Kommunikationssystems zu finden. Um bei der Steuerung der Roboterbiene Kollisionen mit anderen Bienen und dadurch aggressivem Verhalten auszuweichen, wurde eine Kamera an den Roboter angebracht und bestimmte Regionen wurden mit einem Vision System untersucht. Dadurch werden näher kommende Bienen lokalisiert und durch eine Positionskorrektur der Roboterbiene Kollisionen verhindert.

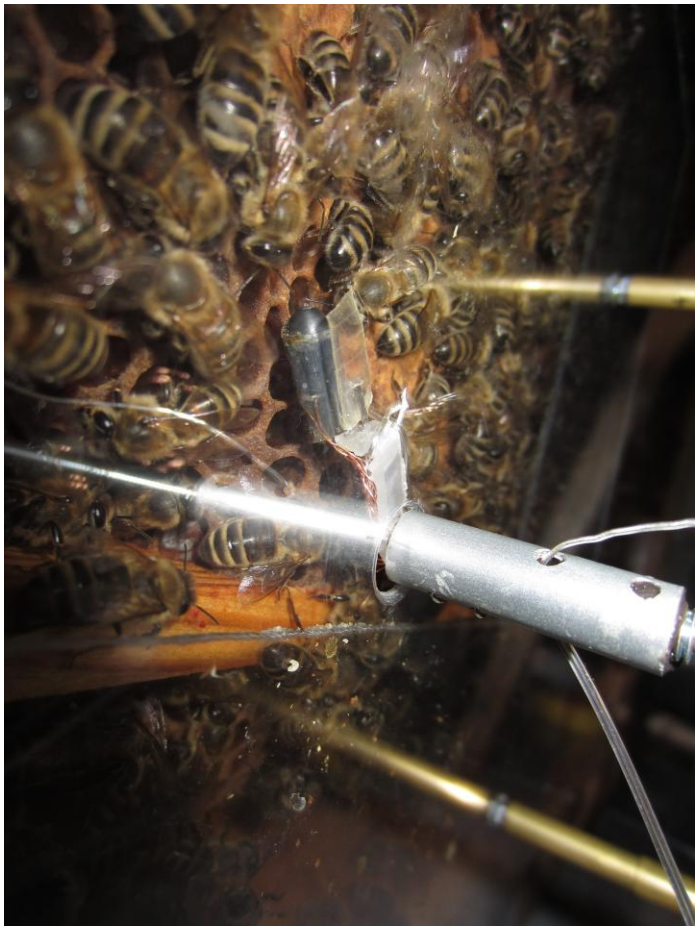

Abbildung 4 [18]: Der finale Prototyp der an der Freien Universität entwickelten Roboterbiene. Das Bienen-Replikat (sichtbar in der Mitte zwischen den Bienen) wird vom Roboter (rechts im Bild) gesteuert.

Ein anderes Beispiel zeigt die erfolgreiche Interaktion von Robotern mit Kakerlaken [6] (Abbildung 5). Der Kakerlakenroboter soll helfen, Schwarmverhalten von Kakerlaken zu verstehen [6]. Er soll sich in eine Gruppe von Kakerlaken integrieren und mit ihnen interagieren [6]. Hierbei werden zur Auswertung der Roboter und lebende Kakerlaken mit einem Vision System lokalisiert um bei besonders langen (bis zu dreistündigen) Experimenten auf einfache und effiziente Weise die abgelaufene Strecke der Kakerlaken und dadurch eine Korrelation mit dem Roboter feststellen zu können [6].

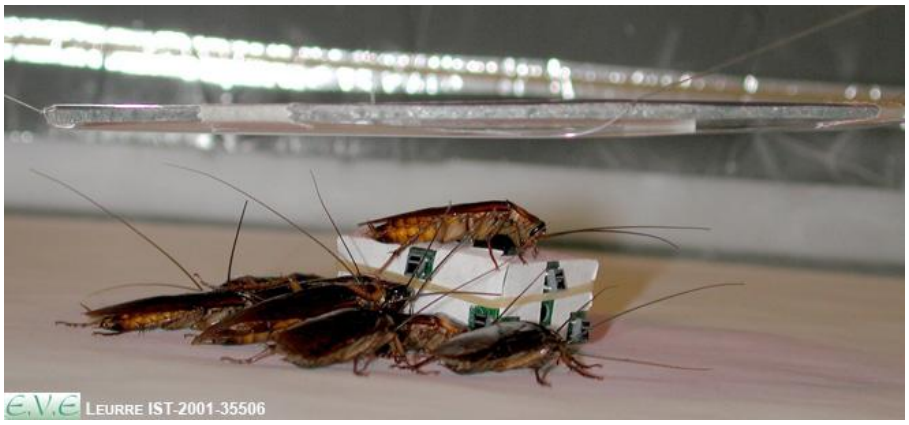

Abbildung 5 [19]: Kakerlaken interagieren mit dem Roboter.

2 Versuchsaufbau

2.1 Fisch-Replikate

Im Rahmen dieser Diplomarbeit wurden in enger Zusammenarbeit mit Biologen zwei verschiedene Prototypen eines Fisch-Replikates entwickelt. Wie in Abschnitt 1.2.1 erläutert, war ein Trackingsystem für die korrekte Steuerung des Roboters notwendig. Da das Fisch-Replikat magnetisch an den Roboter gekoppelt wird, bot es sich an, mit einer oberhalb des Aquariums positionierten Kamera die Nachbildung zu lokalisieren und zu tracken. Dabei musste diese eindeutig von den lebenden Fischen unterscheidbar sein und durfte gleichzeitig in ihrer Morphologie keine großen Abweichungen zu den Fischen darstellen. Es war z.B. nicht möglich, einen großen roten Punkt auf den Rücken des Fisch-Replikates zu kleben und dieses im Bild zu suchen, da das unter Umständen die lebenden Fische abschrecken und dadurch die Interaktion mit dem Replikat stören könnte. Die Biologen werden die Experimente mit zwei verschiedenen Fischarten durchführen: Einerseits mit Stichlingen (Abbildung 1) und andrerseits mit Guppys. Diese Fischarten können kein Licht aus dem Infrarotlichtbereich sehen, daher bot es sich an, Infrarotlicht-LEDs zu verwenden. Für beide Fischarten sind die Arbeitsschritte zur Herstellung identisch.

Die von den Biologen verwendeten Replikate folgen der Vorlage von Ward et al. [3][7][8] und bestehen aus chromatischem Alginat gefüllt mit Hartgips. Die Alginatform wird aus einem toten Fisch erstellt [3]. Er wird hierfür kurz in das Alginat getunkt, bis es seine Form annimmt [3]. Anschließend wird Hartgips in die Vorlage eingegossen [3]. Die Schwanzflosse besteht aus einem Azetatblatt und wird am Gips angeklebt. Dieser wird danach mit Acrylfarben entsprechend bemalt, um die Tönung der Fische bestmöglich zu imitieren [3] (Abbildung 6a). Das Replikat wird am Bauchbereich mit Hilfe eines Kapillargefäßes (15x1,8 mm) mit seiner magnetischen Basis verbunden [3]. Die Basis besteht aus Plexiglas (7x7x5 mm) auf dem zwei Neodym-Magnete mit verschiedener Polung (10x2,5x2,25 mm) befestigt sind [3]. Die Basis wird weiß angemalt, um zum weißen Boden des verwendeten Aquariums zu passen [3]. Dadurch wird die Wahrscheinlichkeit verringert, dass die Basis die Interaktion zwischen den Fischen und dem Roboterfisch stört.

Das Replikat wird durch einen Elektromagneten unter dem Aquarium gesteuert [3]. Dieser ist an einem Plotter angebracht, der sich zweidimensional auf Schienen bewegen kann [3]. Der Plotter wird mit Hilfe eines Computers programmiert [3] und durch diesen das Fisch-Replikat gesteuert (Abbildung 6b). Die Orientierung des Replikates wird automatisch durch die Kopplung der Magnete vorgegeben und nicht aktiv durch den

Plotter gesteuert, so dass die Vorderseite der Nachbildung immer in die Schwimmrichtung schaut. Der Roboterfisch orientiert sich automatisch, da der Elektromagnet die gleiche Polung des Magneten an der Vorderseite der Nachbildung hat (siehe Abschnitt 2.1.1).

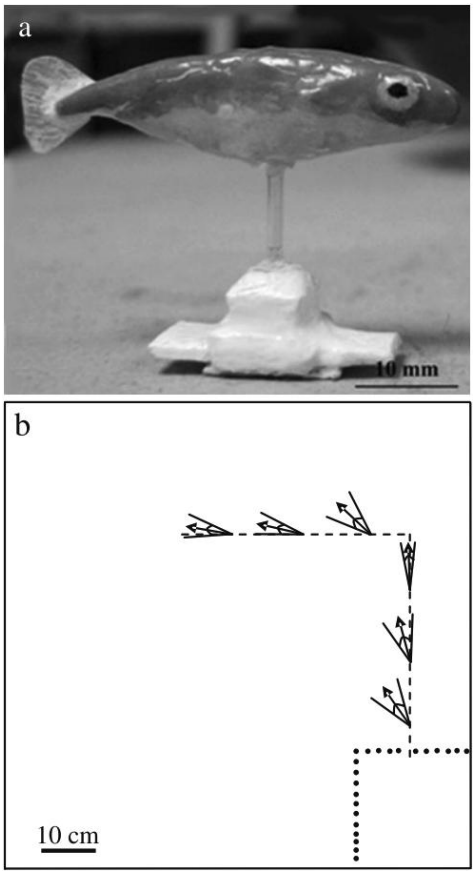

Abbildung 6 [3]: a) Foto des verwendeten Fisch-Replikates. b) Schema des verwendeten Aquariums. Die gepunkteten Striche stellen das Refugium dar, in welchem sich die Fische und das Replikat zu Beginn eines Experimentes befinden. Die gestrichelte Linie stellt die Trajektorie des Replikates dar. Die eingezeichneten Pfeile zeigen, wie einzelne Fische der Nachbildung folgen.

2.1.1 Prototyp 1: Replikate mit einer integrierten Infrarotlicht-LED

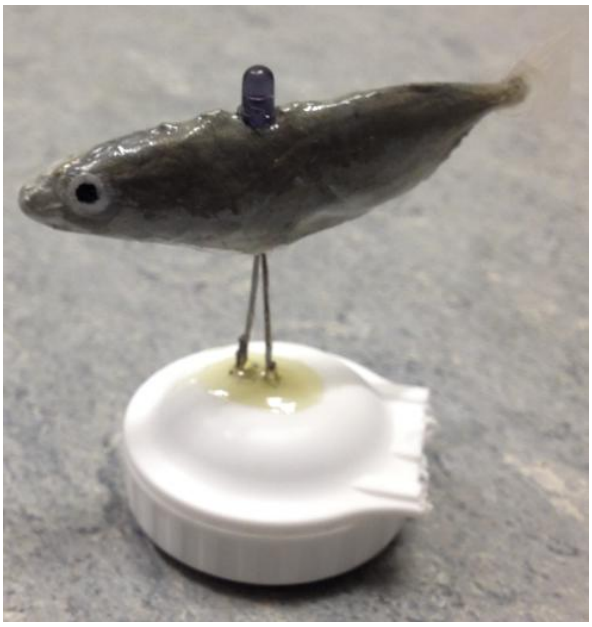

Abbildung 7: Der erste Prototyp des (Stichling-)Fisch-Replikates. Auf dem Rücken ist eine Infrarotlicht-LED zu erkennen. Die Batterie befindet sich in der wasserdichten Basis, die aus einem weißen Kontaktlinsenbehälter besteht. Die Anode und Diode der LED dienen als Verbindungsstück zwischen Nachbildung und Basis und sind innerhalb dieser mit einem Knopfzellenbatteriehalter verbunden.

Für den ersten Prototyp wurde das von den Biologen entworfene Fisch-Replikat modifiziert. Hierfür wird in der Mitte der Nachbildung ein Loch mit einem Durchmesser von 2 mm gebohrt und eine Infrarotlicht-LED (Durchmesser: 3 mm, Wellenlänge: 940 nm) bis zum Anschlag durchgesteckt (Abbildung 8).

Abbildung 8: Zu sehen ist ein Fisch-Replikat, in dem ein Loch mit 2 mm Durchmesser gebohrt wurde.

Um die LED mit Strom zu versorgen werden die Anode und Diode der LED in einem wasserdichten Behälter mit einem Knopfzellenbatteriehalter (Typ 2025) verbunden (Abbildung 7) und dienen damit gleichzeitig als Verbindungsstück zur Basis (Füße). Sie werden mit Epoxy Kleber stabilisiert, der auch für eine Isolation zwischen Wasser und Basis sorgt. Anschließend werden die Füße mit Silikon-Spray eingesprüht, damit sie im Wasser keinen Strom mehr leiten. Für die wasserdichte Basis wird ein einzelner weißer Kontaktlinsenbehälter (Durchmesser: 3 cm, Höhe: 2 cm) verwendet. Der Boden des Tanks, den die Biologen verwenden um ihre Experimente durchzuführen, ist weiß. Dadurch, dass der Kontaktlinsenbehälter dieselbe Farbe wie der Boden hat, wird die

Wahrscheinlichkeit reduziert, dass dieser von den lebenden Fischen als Fremdkörper erkannt wird und die Interaktion stört. Im Kontaktlinsenbehälter ist zusätzlich ein entsprechender Widerstand zwischen LED und Knopfzellenbatteriehalter in Reihe geschaltet. Um die LED anzuschalten, muss eine Knopfzellenbatterie (Typ 2025, 3V) in den Knopfzellenbatteriehalter gesteckt werden.

Um das Replikat im Wasser zu steuern werden zwei Blockmagnete mit verschiedener Polung (B x L x H: 5 x 2,5 x 1,5 mm) so an die Unterseite des Kontaktlinsenbehälters geklebt, dass bei verschlossener Basis Magnet A die vordere Seite des Replikates repräsentiert und Magnet B die hintere Seite. An den Roboter wird ein starker Magnet mit der Polung von Magnet A befestigt (Abbildung 9). Das führt dazu, dass sich das Replikat automatisch in die Fahrtrichtung des Roboters dreht. Um die Reibung zwischen der Nachbildung und dem Boden des Aquariums zu reduzieren, werden an die Magnete kleine Plastiknoppen geklebt.

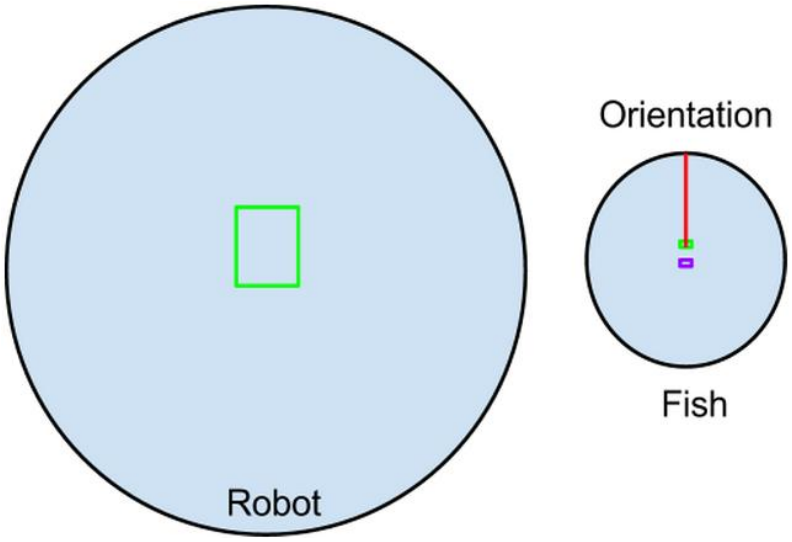

Abbildung 9: Links befindet sich ein Schema des Roboters, rechts des Fisch-Replikates. In der Mitte befinden sich jeweils die Blockmagnete. Die Polung (farblich gekennzeichnet) zwischen dem Magneten am Roboter und dem Magneten an der Vorderseite des Replikates muss übereinstimmen damit sich der Prototyp automatisch in die gleiche Richtung wie der Roboter bewegt.

Dieser Prototyp wurde aus zwei Gründen schnell verworfen: Einerseits konnte hier im Falle einer 180° Richtungsänderung die „zackige“ Bewegung (siehe Abschnitt 1.2) nicht vermieden werden. Andrerseits wurde festgestellt, dass die Orientierung des Roboters für eine korrekte Positionsregelung eventuell unabdingbar ist (siehe Abschnitt 3.4.3).

2.1.2 Prototyp 2: Replikate mit zwei integrierten Infrarotlicht-LEDs

Abbildung 10: Der zweite Prototyp mit Basis. Oben sind zwei Infrarotlicht-LEDs zu sehen, die leicht aus dem Oberkörper des Replikates herausragen. Aus dem Unterkörper dringen zwei Kupferstangen, die in der Basis stecken, heraus. Diese dienen als Verbindungsstück zur Basis und leiten gleichzeitig den Strom in die LEDs.

Der neue Prototyp 2 verwendet dieselbe Basis wie Prototyp 1 (siehe Abschnitt 2.1.1), mit dem Unterschied, dass der Widerstand im Fisch-Replikat statt in der Basis verbaut wird. Dadurch wird im Behälter mehr Platz für den Batteriehalter geschaffen.

Abbildung 11: Die aufgeschraubte Basis. Im inneren ist der verwendete Knopfzellenbatteriehalter zu erkennen.

In das Replikat werden zwei Infrarotlicht-LEDs verbaut. Da das nachträgliche Verbauen von zwei LEDs mit einem in Reihe geschalteten Widerstand sehr aufwendig bis unmöglich ist, wurde zusammen mit den Biologen entschieden, dass diese bei der Erstellung der Replikate direkt in das chromatische Alginat gesteckt und dann umgipst werden. Die beiden verbauten LEDs haben einen Abstand von genau 2 cm. Ein größerer Abstand ist nicht möglich, da die kleinsten verwendeten Fisch-Replikate eine Länge von 4 cm haben und die beiden LEDs nicht mehr reinpassen würden. Ein kleinerer Abstand ist theoretisch möglich, wurde aber im Rahmen dieser Arbeit nicht getestet. (Es wurden keine SMD-LEDs verwendet, da diese die Herstellung des Prototyps unnötig erschweren würden.) Zwischen den beiden LEDs befindet sich der entsprechende Widerstand. Als Verbindungsstück zwischen LEDs und Basis werden statt Anode und Diode zwei Kupferstangen (Durchmesser: 0,8 mm, Länge: 4 cm), die entsprechend Abbildung 12 mit den beiden LEDs verlötet werden, verwendet.

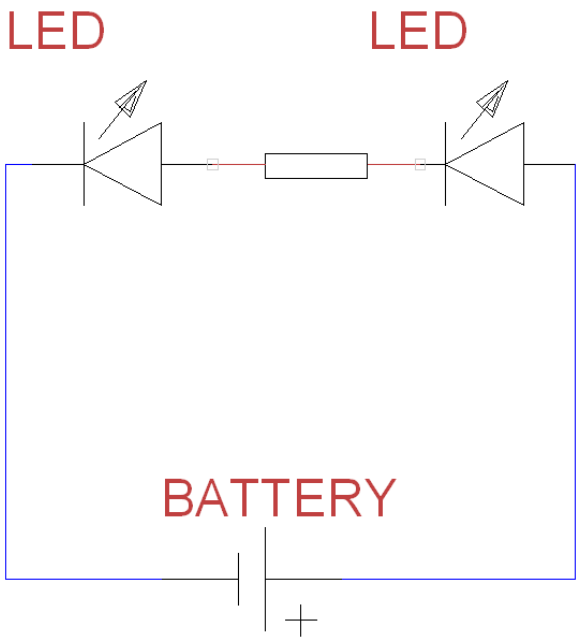

Abbildung 12: Die Schaltungsskizze für die Elektronik in Prototyp 2. Unten befindet sich die Stromquelle (Basis). Die dort eintreffenden blauen Linien stellen die beiden Kupferstangen dar.

Für die Kopplung des Replikates mit dem Roboter werden wie in Abschnitt 2.1.1 Blockmagnete an den Boden der Basis geklebt. Da das Replikat dieselbe Orientierung wie der Roboter haben soll, müssen dessen Magnete wie in Abbildung 13 angebracht werden. Der Roboter wird entsprechend um einen weiteren Magneten erweitert.

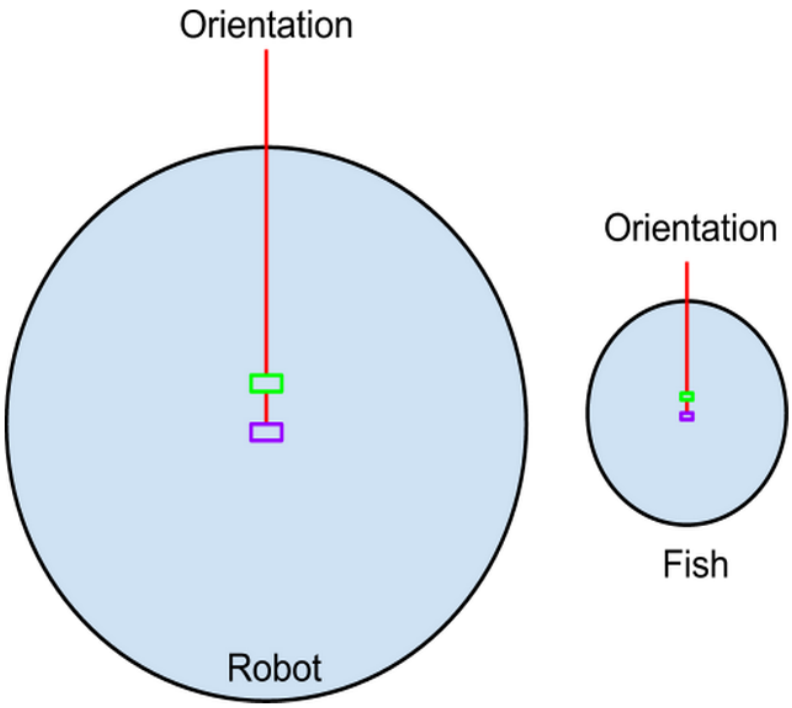

Abbildung 13: Links ist ein Schema des Roboters zu sehen, rechts des Fisch-Replikates. In der Mitte befinden jeweils zwei verschieden gepolte Blockmagnete. Die Polung zwischen den beiden Magneten an der Vorderseite (grün) und den beiden Magneten an der Hinterseite (lila) muss für eine korrekte Steuerung übereinstimmen.

2.2 Experimentelle Umgebung

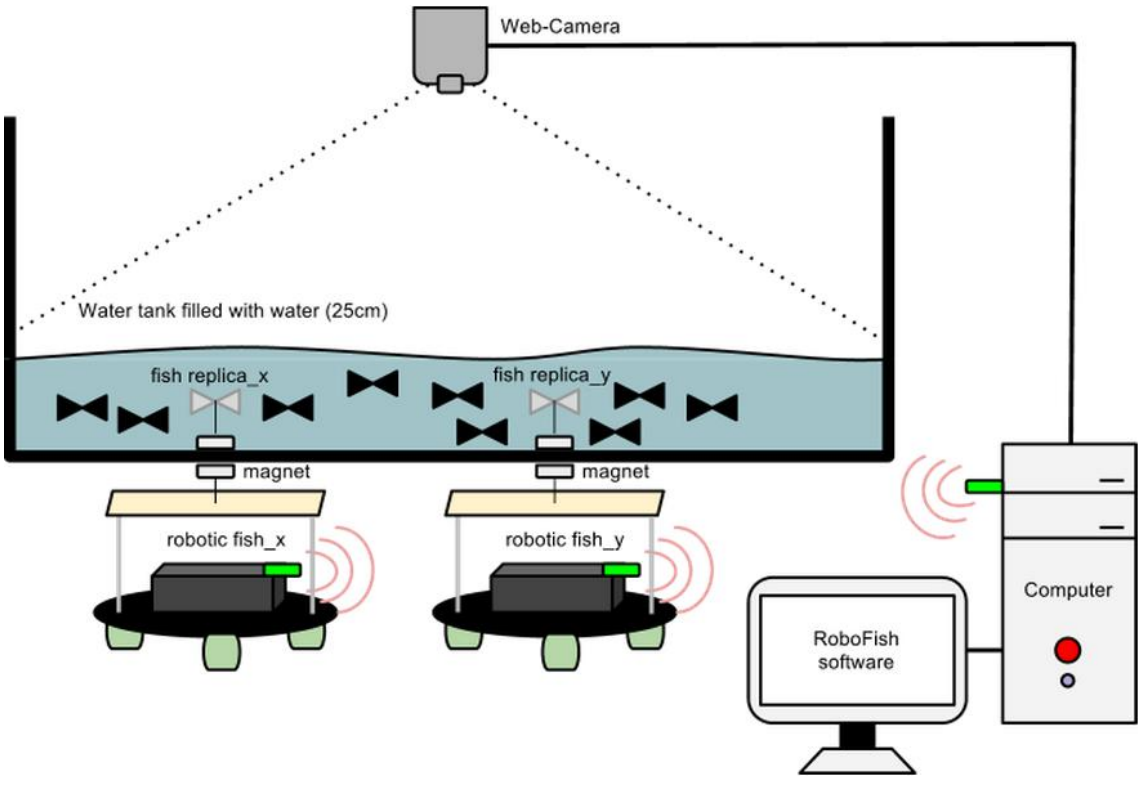

Abbildung 14 (Hai Nguyen): Der verwendete Versuchsaufbau. Im Aquarium befinden sich die Fisch-Replikate. Roboter (Durchmesser: 17 cm) steuern unter dem Aquarium durch magnetische Kopplung die

Nachbildungen. 1,30 Meter über dem Aquarium hängt eine USB Kamera, mit der Bilder an einen Computer weitergeleitet werden. Auf dem Computer läuft die RoboFish 2.0 Software, welche für die Kommunikation mit den Robotern zuständig ist und die mit Hilfe der Kamerabilder das Tracking übernimmt.

Um sehr genaue Testergebnisse und eine realitätsnahe Simulation erzielen zu können, wurde eine Versuchsumgebung gebaut, welche die Laborumgebung der Biologen möglichst genau nachstellt (Abbildung 14). Es ist z.B. wichtig, das Tracking und die Bewegungen der Fisch-Replikate im Wasser testen zu können da Wasser das zum Tracking verwendete Infrarotlicht absorbiert und daher Wasserwellen die Positionsbestimmung stören können.

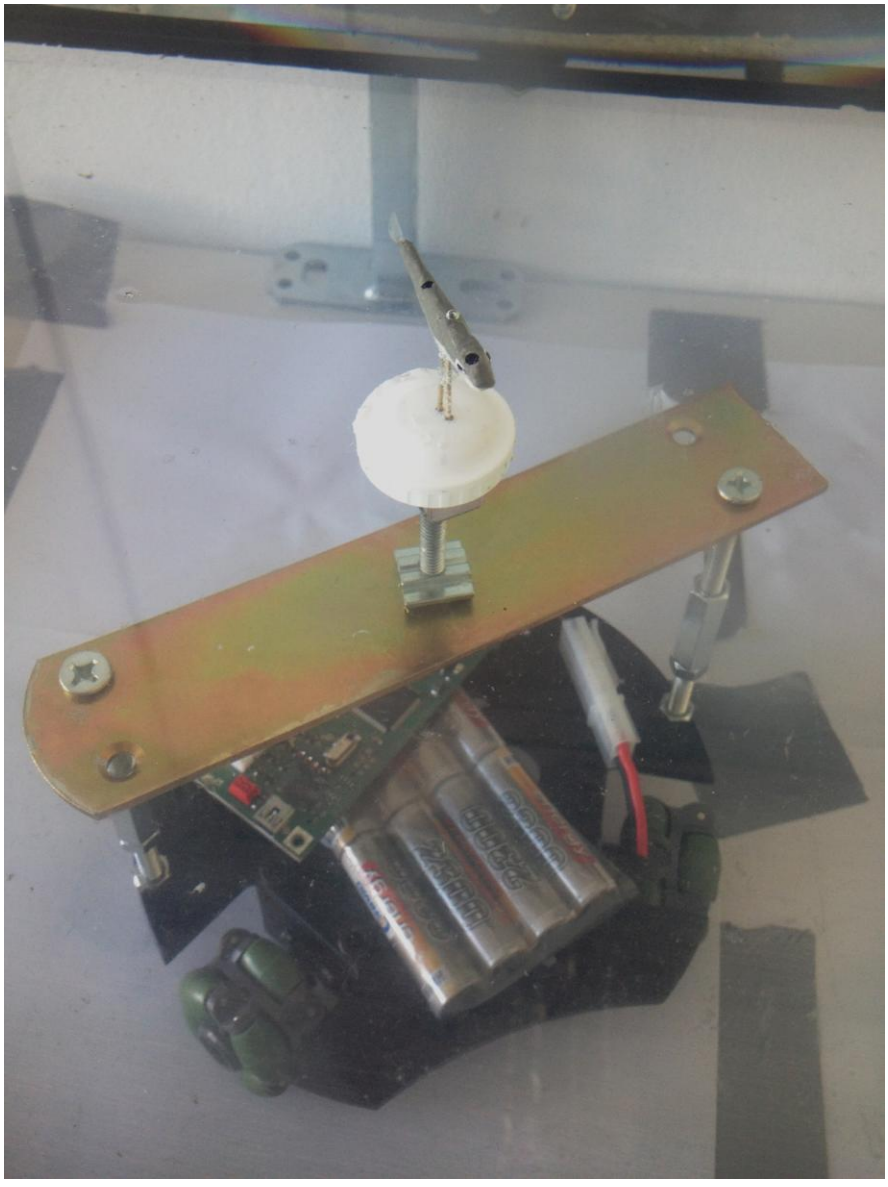

Abbildung 15: Unten ist der Roboter zu sehen, oben die Nachbildung im Wasser. Das Fisch-Replikat ist magnetisch an den Roboter gekoppelt. Beide haben dieselbe Orientierung.

Ein mit 25 cm Wasser gefülltes 50x30x40 cm Aquarium wird verwendet, um das Fisch-Replikat unter Wasser testen zu können (Abbildung 14). Es wird diese Wasserhöhe benutzt, damit lebende Fische in den Experimenten der Biologen nicht simultan übereinander schwimmen und sich dadurch nicht kreuzen können. Mögliche Überkreuzungen würden die Untersuchungen stören, da dann nicht mehr eindeutig bekannt wäre, welcher Fisch welchem folgt. Eine handelsübliche USB Kamera wird in 1,30 m Höhe über dem Aquarium gehängt um den gesamten Bereich der Trajektorie abzudecken (Abbildung 14). Das Aquarium wird in eine 1x1 m Holzplatte montiert, die mit in der Höhe verstellbaren Füßen auf einem ebenen Boden steht (Abbildung 16).

Abbildung 16: Die Holzplatte, in der das Aquarium eingehängt wurde. Unter dem Aquarium ist jetzt genügend Platz, um dort den Roboter zu platzieren und zu steuern.

Unter dem Aquarium befindet sich ein omnidirektional fahrender Roboter mit einem Durchmesser von 17 cm. Das Replikat wird magnetisch an den Roboter mittig gekoppelt, hat in dieselbe Orientierung wie der Roboter (Abbildung 15, Abbildung 17, siehe Abschnitt 2.1) und besitzt zwei nach oben gerichtete Infrarotlicht-LEDs, die zur Lokalisierung verwendet werden. Auf einem Computer läuft die RoboFish 2.0 Software, die das Tracking übernimmt und den Roboter steuert.

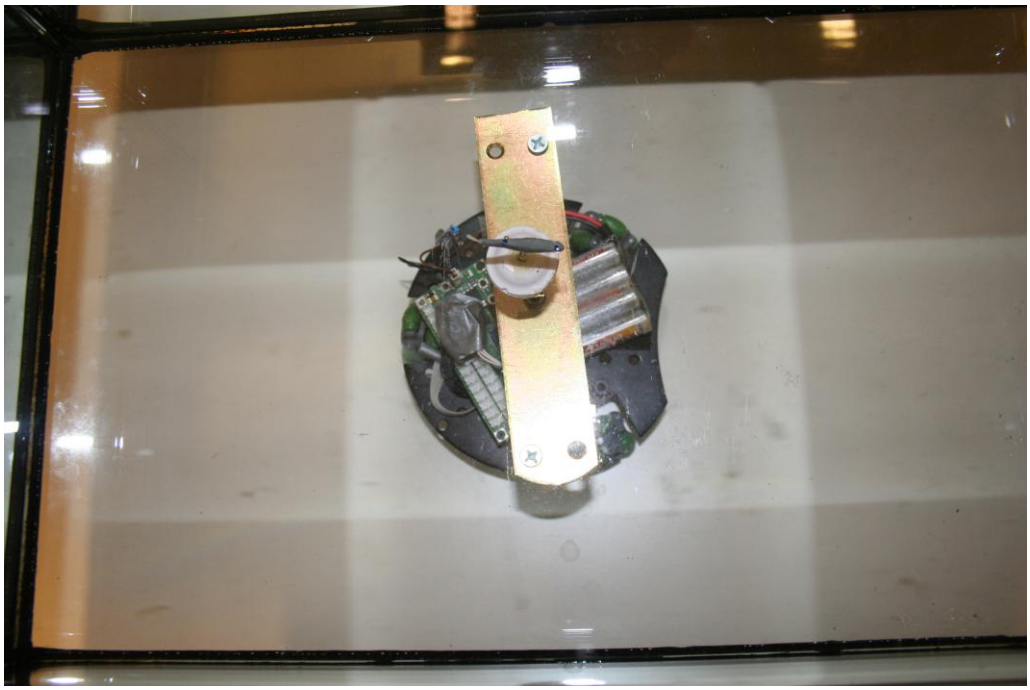

Abbildung 17: Das Aquarium aus der Sicht von oben. Das Fisch-Replikat befindet sich im Wasser. Unter dem Aquarium befindet sich der Roboter, der die Nachbildung steuert.

Die Kamera wird hier oberhalb des Aquariums montiert, um die größtmögliche Flexibilität beim Aufbau zu gewährleisten. Um den gesamten Bereich tracken zu können ist eine entsprechende Kamerahöhe notwendig. Würde die Kamera unterhalb des Roboters positioniert werden und würde statt der Replikate der Roboter getrackt werden, so müsste das Aquarium unter Umständen sehr hoch stehen, was bei einem größeren Becken problematisch werden könnte und das System in seiner Flexibilität einschränkt.

2.3 Kamera

Für das Tracking wird eine handelsübliche USB Kamera benutzt (Abbildung 18), in diesem Fall eine „Hercules Deluxe Cam“. Um eine größtmögliche Effizienz beim Tracking zu gewährleisten muss die Kamera im Normalbetrieb eine hohe Framerate (mindestens 20 FPS) bei einer Auflösung von 640x480 liefern. Die Belichtungszeit der Kamera muss manuell über den Hardwaretreiber einstellbar sein.

Abbildung 18: Die für das Tracking verwendete Kamera.

An der Kamera werden hardwaretechnische Modifikationen vorgenommen, um das Bild für das Tracking zu optimieren.

Abbildung 19: Eine standardisierte m12 Linse. Fixierfeld: 92°. Die Linse hat keinerlei Filter.

Hierfür wurde eine neue standardisierte m12 Linse, die keinerlei Filter hat und eine möglichst schwache Verzerrung liefert (Abbildung 19), gekauft und an die Kamera montiert. Die Linse hat ein 92° Fixierfeld. Der Kamerawinkel beträgt 90° und der Mittelpunkt des Kamerabildes gleicht dem Mittelpunkt des Aquariums (Abbildung 20).

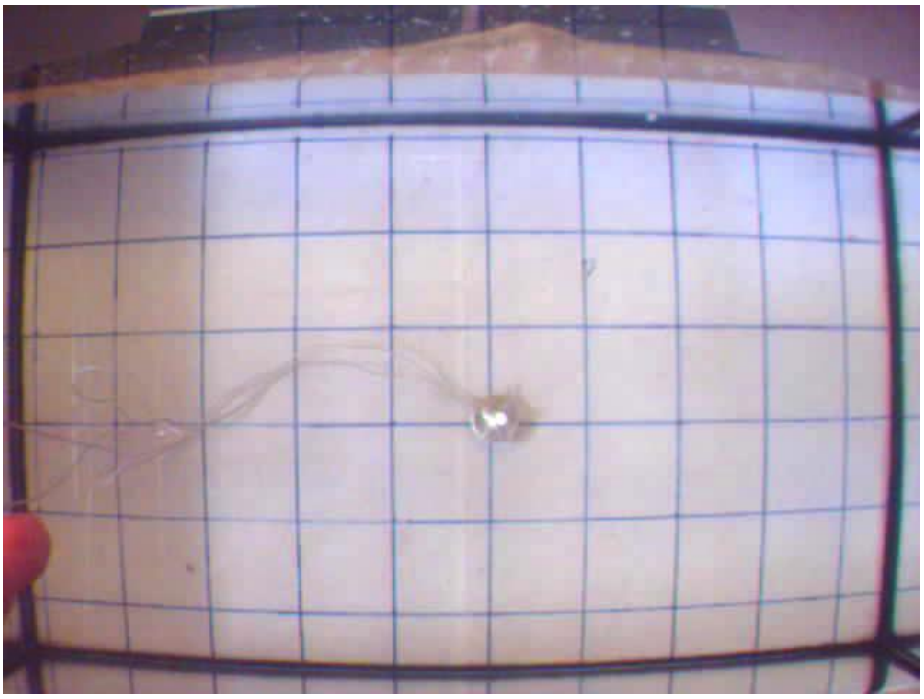

Abbildung 20: Kamerabild ohne Modifikationen bei guter Raumhelligkeit. Mittig ist eine leuchtende Infrarotlicht-LED zu erkennen.

Abbildung 21: Dieselbe Linse aus Abbildung 19 mit einem aufgeklebten Infrarotlicht-Passfilter.

Da Infrarotlicht im 940 nm Wellenbereich lokalisiert werden soll (siehe Abschnitt 2.1) ist es hilfreich, wenn alle anderen Wellenlängen durch einen Filter neutralisiert werden. Hierfür wurde ein Infrarotlicht-Passfilter auf die hintere Seite der Linse geklebt (Abbildung 21). Das Kamerabild besteht nach aufkleben des Filters bei optimal eingestellter Belichtungszeit (siehe Abschnitt 2.3.1) nur noch aus schwarzen und weißen Pixeln (Abbildung 22) und ist somit optimiert, um softwareseitig tracken zu können. Da

diese Modifikationen hardwareseitig sind, werden Performanz-technische Einbußen gespart. Es müssen keine Algorithmen implementiert werden, die sich softwareseitig im Tracking um diese Modifikationen kümmern und zeitaufwendig sein könnten.

Abbildung 22: Kamerabild mit Infrarotlicht-Passfilter bei guter Raumhelligkeit. Eine klar leuchtende Infrarotlicht-LED ist zu sehen. Die anderen Lichtstrahlen werden herausgefiltert.

2.3.1 Belichtungszeit

Die Belichtungszeit gibt an, wie lange ein Bild dem Licht ausgesetzt wird. Eine große Belichtungszeit bedeutet, dass mehr Licht vom Aufnahmemedium aufgenommen wird. Es lässt sich hier eine Analogie zum menschlichen Auge herstellen: Blickt ein Mensch in einer sternenklaren Nacht zuerst lange in eine sehr helle Lampe, dann verengt sich seine Pupille und nimmt dadurch weniger Licht auf. Die Lampe strahlt so hell, dass er sie trotz kleiner Pupillengröße, also einer kleineren Aufnahmefläche, gut erkennen kann. Schaut er nun direkt danach in den Sternenhimmel, so erkennt er zuerst nur die sehr hellen Objekte wie z.B. den Mond und den Mars. Je länger er ungestört in den Himmel schaut, desto mehr Sterne werden für ihn sichtbar. Nach einer bestimmten Zeit hat sich seine Pupille maximal geöffnet und lässt mehr Licht auf das Auge treffen, wodurch er mehr erkennen kann. Diese Anpassung geschieht unterbewusst, der Mensch hat seine eigene Belichtungszeit erhöht.

Die Belichtungszeit der Kamera muss in Abhängigkeit vom verwendeten Tracking-Algorithmus (siehe Abschnitt 3.4) korrekt eingestellt werden. Wie in Abschnitt 3.2.2 beschrieben, wird bei der Binarisierung von einem Bild ausgegangen, das aus sehr dunklen und sehr hellen Pixeln besteht. Wird die Belichtungszeit der Kamera auf einen sehr hohen Wert gestellt, so trifft mehr Licht auf die Linse ein. Befindet sich die Kamera in einem Raum mit Fenstern, dann dringt eventuell Sonnenlicht in den Raum. Sonnenlicht besteht zu einem Großteil aus Infrarotlicht. Bei einer hohen Belichtungszeit

kann dieses Licht trotz Infrarotlicht-Passfilter erfasst werden. Das Bild besteht im ungünstigsten Fall aus mehreren Objekten deren Helligkeit den in der Binarisierung gesetzten Schwellenwert überschreiten und somit als LED erkannt werden (Abbildung 23). Außerdem können zwei LEDs zu einer verschmelzen und dadurch nicht mehr unterschieden werden (Abbildung 23).

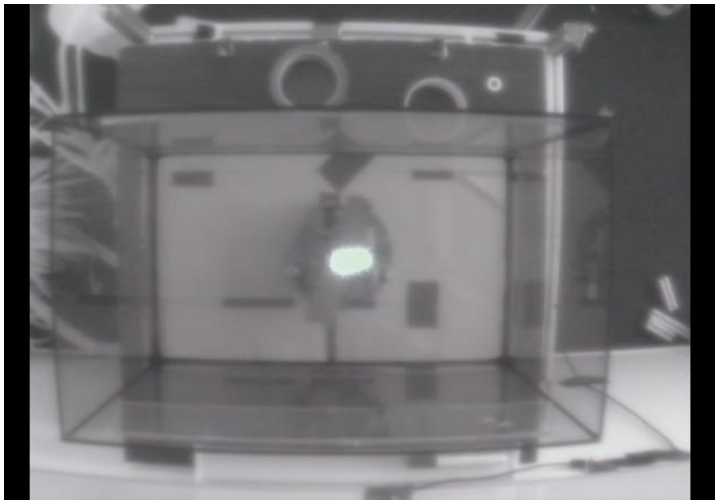

Abbildung 23: Das Kamerabild mit hoher Belichtungszeit. Die gesamte Versuchsumgebung ist sichtbar. In der Mitte leuchtet ein Fisch-Replikat mit zwei eingebauten LEDs, die zu einer verschmolzen sind (Prototyp 2, siehe Abschnitt 2.1.2). Eine hohe Belichtungszeit führt dazu, dass die in Abschnitt 2.3 vorgestellten hardwaretechnischen Modifikationen wirkungslos werden.

Wird die Belichtungszeit verringert, lässt sich in der Mitte das Fisch-Replikat mit zwei eingebauten deutlich unterscheidbaren LEDs erkennen. Der Rest des Bildes ist schwarz (Abbildung 24).

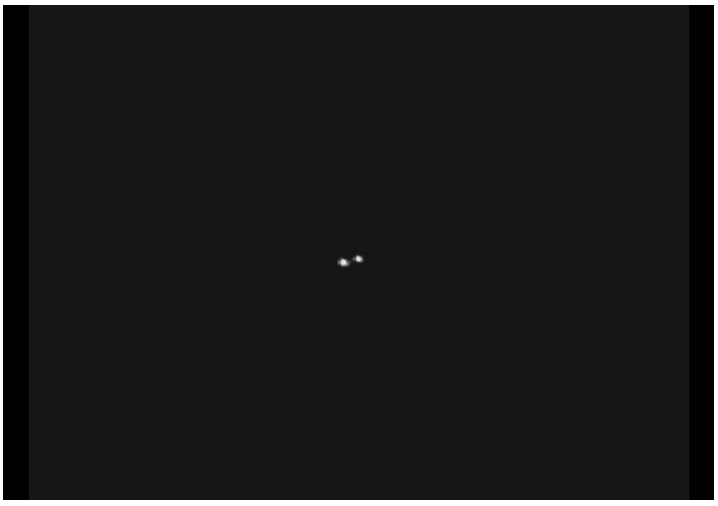

Abbildung 24: Das gleiche Kamerabild aus Abbildung 23 mit geringer Belichtungszeit (fast null). Mittig ist ein Fisch-Replikat mit zwei eingebauten und klar unterscheidbaren LEDs (Prototyp 2, siehe Abschnitt 2.1.2) zu erkennen.

Es wird eine möglichst geringe Belichtungszeit gewählt, um die oben beschriebenen Problematiken zu umgehen.

Steht der Versuchsaufbau, so wird die Belichtungszeit einmalig festgelegt und über den Kameratreiber entsprechend konfiguriert.

3 Tracking-System

Im Rahmen dieser Diplomarbeit wurde eine Echtzeit-basierte Tracking-Klasse für die RoboFish 2.0 Software geschrieben und getestet. Die Tracking-Klasse kann beliebig viele Roboterfische lokalisieren und bietet dem Nutzer eine Vielzahl an Optionen zur Modifikation der Lokalisierungsparameter. Es wurden in Zusammenarbeit mit Hai Nguyen, der für die Programmierung des Roboters (siehe Abschnitt 1.2 und Abschnitt 2.2) zuständig war, Schnittstellen für die Verwendung der Klasse definiert. Im Folgenden werden zuerst die angewandten Algorithmen vorgestellt, im Verlauf aufgetretene Probleme erläutert und deren Lösungen beschrieben und es wird erklärt, wie die Klasse von einem Benutzer zu verwenden ist.

3.1 Entwicklungsumgebung und Software

Zum Programmieren wird als Entwicklungsumgebung Eclipse (v. 3.7.2) benutzt. Die verwendete Programmiersprache ist C++. Für die Bildverarbeitung wird die frei verfügbare OpenCV Library (v. 2.3.1) verwendet, die eine Vielzahl an Bildbearbeitungs-Algorithmen bereitstellt. Zur einer grafischen Benutzeroberfläche zum Testen der Klasse (siehe Abschnitt 3.5) wird die frei verfügbare Qt-Library (v. 4.8) benutzt. Es wird auf einem Computer mit einem Intel Core 2 Duo (3 GHz), 4GB Arbeitsspeicher und Windows 7 (64-Bit) programmiert und getestet.

3.2 Kamerabild

OpenCV wird verwendet, um ein zweidimensionales Bild mit einer Auflösung von 640x480 Pixeln von der Kamera zu erhalten. Es ist möglich, die Auflösung bei Bedarf zu ändern. Das Bild der Kamera wird zeilenweise eingelesen und dann zu einer Matrix zusammengefasst. Die Matrix enthält die einzelnen Farbwerte für jeden Pixel. Abbildung 25 verdeutlicht, wie ein Bild in OpenCV aufgebaut ist.

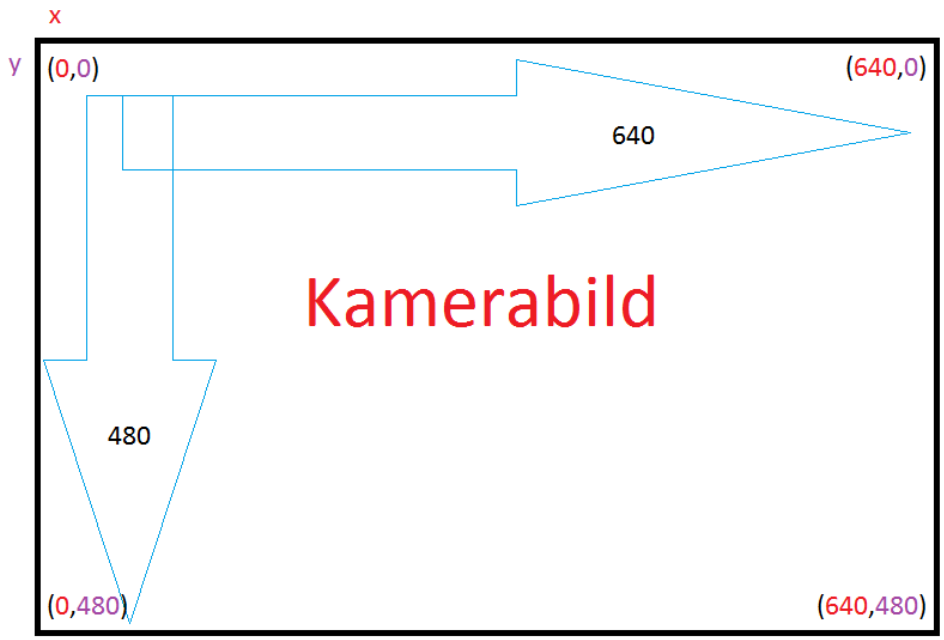

Abbildung 25: Aufbau des Kamerabildes in OpenCV.

Das Kamerabild wurde (siehe Abschnitt 2.3) hardwaretechnisch bereits vorbearbeitet (Abbildung 22), muss jedoch softwaretechnisch für das Tracking weiter optimiert werden. Die folgenden Optimierungen werden im Tracking in jedem Frame angewendet.

3.2.1 Entzerrung

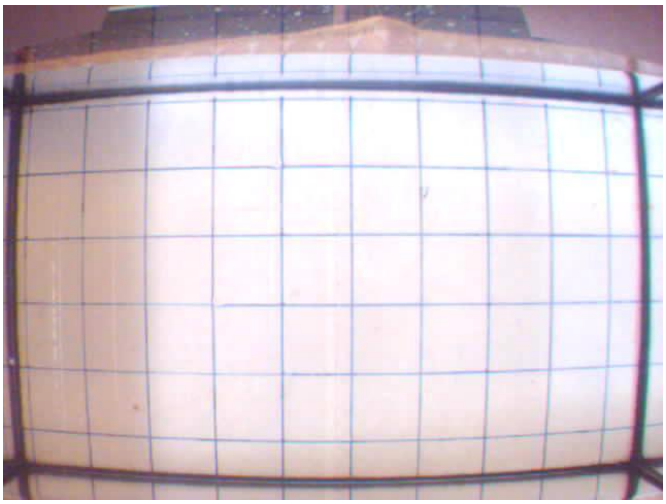

Abbildung 26: Verzerrtes Kamerabild. Die Quadrate an den Bildrändern sind kleiner als die in der Mitte.

In Abbildung 26 ist eine Verzerrung zu erkennen, die durch die Linse erzeugt wird. Lochkameras weisen immer aufgrund der Unlinearität der Optik eine minimale radiale Verzerrung auf [2]. Es gibt zwei Arten von radialen Verzerrungen: Tonnenförmige Verzerrungen und kissenförmige Verzerrungen [2]. Das Aquarium hat eine quadratische

Form. Da in die gezeichneten Quadrate, die in Abbildung 26 unter dem Aquarium liegen, an den Bildrändern kleiner werden, wird von einer tonnenförmigen Verzerrung (Abbildung 27) gesprochen [2]. Würden diese größer werden, so würde es sich um eine kissenförmige Verzerrung handeln [2]. In jedem Fall muss die Verzerrung softwareseitig ausgeglichen werden, um in der Positionsbestimmung Ungenauigkeiten auszugleichen.

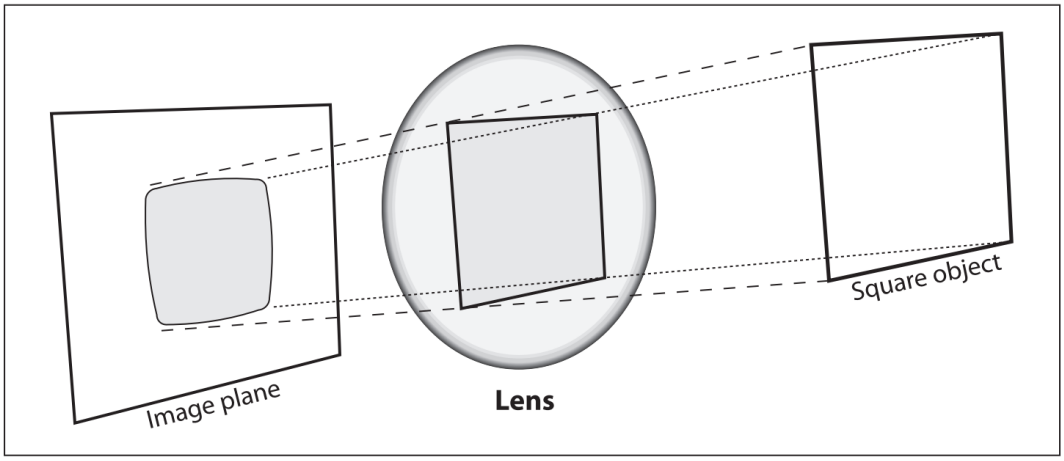

Abbildung 27 [5]: Eine tonnenförmige Verzerrung des aufgenommenen Objektes wird hier durch die Kamera verursacht.

Um das Bild zu kalibrieren und zu entzerren, wurde die Schachbrett-Methode von Gary Bradsky und Adrian Kaehler [2] verwendet. Hierbei werden zwei Matrizen, die intrisische Matrix und die extrinsische Matrix, einmalig berechnet und in jedem Frame für die Entzerrung verwendet. Die extrinsische Matrix definiert den Zusammenhang zwischen der Welt und der Kamera [2], ihre Position und ihre Richtung in Bezug auf ein Weltkoordinatensystem, das sich nicht ändert [2]. Die intrinsische Matrix definiert den Zusammenhang zwischen dem metrischen dreidimensionalen Kamerakoordinatensystem und den zweidimensionalen Bildpixeldaten [2]. Beide Matrizen sind für eine gegebene Linse und Kamera gleich und ändern sich nicht. Es ist daher sinnvoll, diese separat einmalig zu berechnen und dann die Matrizen, hier in einer XML Datei, zu speichern. Es wurde eine Methode geschrieben, die automatisiert das Bild mit Hilfe eines Schachbrettes (Abbildung 28) entzerrt und die beiden Matrizen in einer XML Datei zur späteren Wiederverwendung speichert.

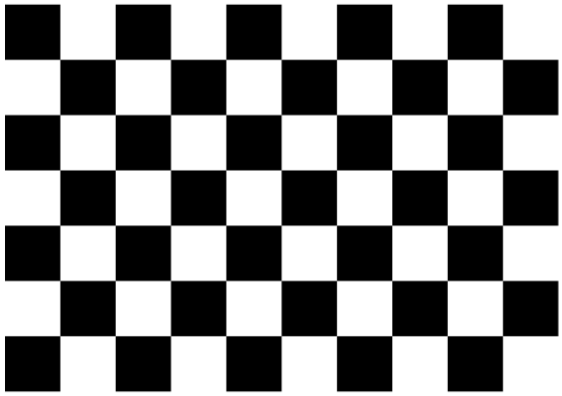

Abbildung 28: Das zur Entzerrung verwendete Schachbrett bestehend aus weißen und schwarzen 10x7 Quadraten.

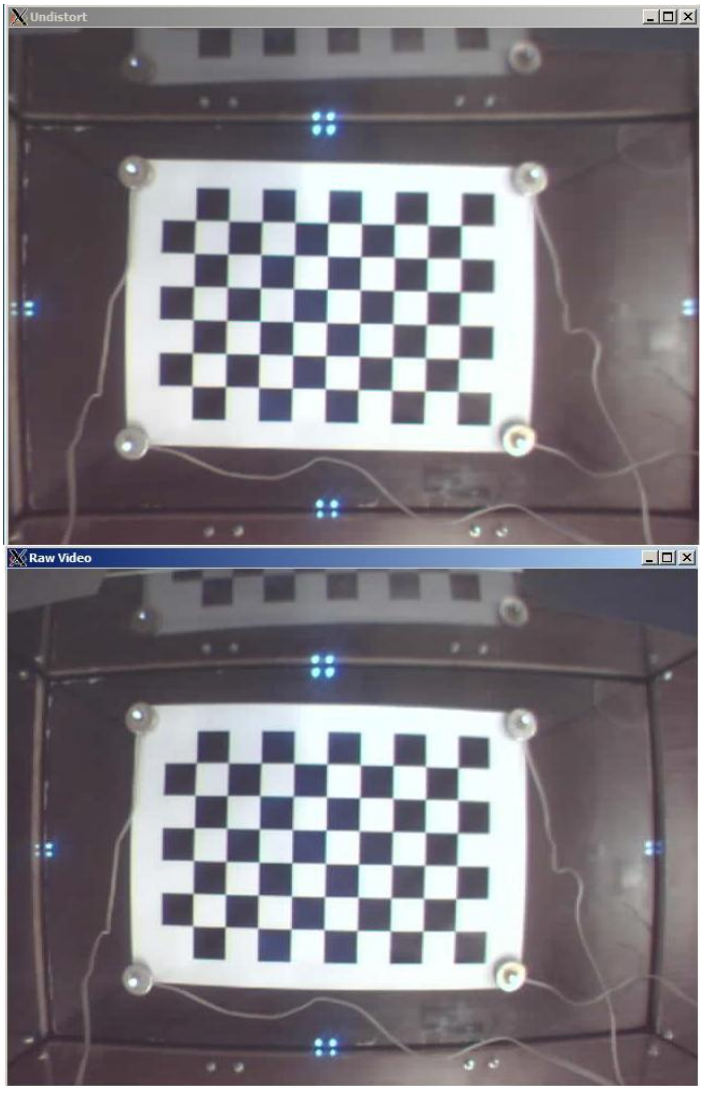

Abbildung 29: Unten ist das unbearbeitete Bild der Kamera zu sehen, oben das entzerrte Bild. Anhand der Seitenränder des Aquarium ist eine geringere Verzerrung im oberen Bild im Gegensatz zum unteren Bild zu sehen.

3.2.2 Binarisierung

Abbildung 22 verdeutlicht, was das Ziel des Trackings ist. Ein Bild bestehend aus dunklen und hellen Pixeln soll nach „besonders“ hellen Pixeln durchsucht und entsprechend einer LED und dann einem Fisch-Replikat zugeordnet werden. Dafür müssen diese „besonders“ hellen Pixel vor der Zuordnung eindeutig festgestellt und gespeichert werden. Die gespeicherten Daten werden durchsucht und die Zuordnungen (statt im Ursprungsbild, das aus verschieden hellen Farben besteht) in diesen getroffen.

Diese „besonders“ hellen Pixel sollen eine LED darstellen. Was bedeutet „besonders“ hell? Ein Pixel besteht aus drei Farbwerten: Rot, grün, blau (RGB) [10]. Jeder dieser Werte gibt an, wie viel Prozent einer Farbe in einem Pixel vorhanden ist [10]. Die Werte gehen von 0 bis 255. Ein hoher Wert bedeutet, dass die Farbe besonders stark vertreten ist [10]. Durch die Mischung dieser drei Farben ergibt sich der tatsächliche Farbwert [10]. Hat ein Pixel den RGB Wert (255,0,0) so heißt das, dass die rote Farbe hundertprozentig vertreten ist und die Farben Grün und Blau gar nicht vorkommen. Der Pixel nimmt damit den Farbwert rot an. Soll die Helligkeit (Luminanz) des Pixels bestimmt werden, ist der Mittelwert der Helligkeiten aller drei Farben nötig [10]. Um die Farben für das menschliche Auge korrekt darzustellen, wird berücksichtigt, welche Farben vom Auge besser gesehen werden, also heller sind [10]. Das Auge sieht grüne Farben besser als rote Farben und rote Farben besser als blaue Farben [10]. Im RGB Farbmodell werden diese Verhältnisse berücksichtigt [10]. So kann ein Mensch das Bild mit korrekt dargestellten Farben erkennen [10]. Daraus lässt sich folgende Schlussfolgerung treffen: Die Helligkeit eines Pixels ist die Summe aller Farbwerte, die jeweils mit einer Präferenz multipliziert werden und anschließend durch die Gesamtanzahl der vorhandenen Farbwerte geteilt werden, also [10]: $helligkeit = \frac{(x*r+y*g+z*b)}{3}$, wobei $x = 0{,}2126$; $y = 0{,}7152$; $z = 0{,}0722$ [10]. Je größer dieser Wert ausfällt, desto heller ist der Pixel.

Um eindeutig feststellen zu können, ob ein Pixel Teil einer LED ist, wird ein Schwellenwert benötigt. Dieser Schwellenwert hängt von verschiedenen Parametern ab (Belichtungszeit der Kamera, Kamerahöhe, Kamerawinkel) und kann vom Benutzer in der Tracking Klasse festgelegt werden.

Bei der Binarisierung wird eine Matrix mit der Größe des Bildes (also der Auflösung der Kamera) initialisiert und dann das Bild zeilenweise analysiert. Dabei wird für jedes Pixel überprüft, ob es gleich oder größer als der festgelegte Schwellenwert ist. Ist dies der Fall, dann wird an der entsprechenden Stelle in der erstellten Matrix eine 1 geschrieben. Sonst eine 0. Wird die Matrix als Bild dargestellt, so sind die Stellen, an denen eine 1 steht, weiße Pixel und die Stellen, an denen eine 0 steht, schwarze Pixel (Abbildung 30).

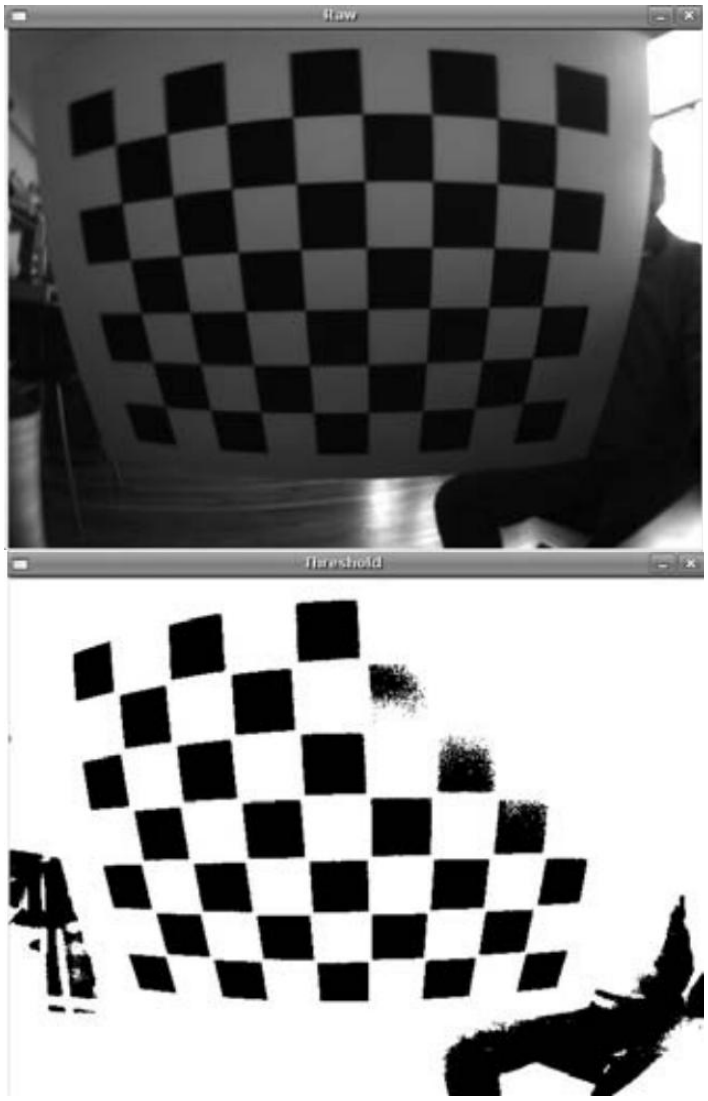

Abbildung 30 [2]: Beispiel einer Binarisierung. Oben das unveränderte Bild, unten das binarisierte Bild. Alles über dem Schwellenwert (hier mit 50 gewählt) ist im binarisierten Bild weiß.

3.3 Rektifizierung

Zwei gleich ebene Objekte, die in zwei verschiedenen Bildern unterschiedlich im Bild positioniert sind, besitzen dieselbe Homographie [2]. Durch die Errechnung dieser lassen sich geometrische Verzerrungen mit Hilfe einer Matrizenmultiplikation ausgleichen [2]. Die Ursachen dieser Verzerrungen sind vielfältig. Sind z.B. planare Objekte auf unebenem Gelände positioniert worden, müssen Unebenheiten ausgeglichen werden.

Dieses Verfahren wird benutzt, um zu vermeiden, dass ein minimal fehlerhafter Kamerawinkel zu großen Messfehlern führt (Abbildung 31). Auch wird durch eine Rektifizierung zusätzlich der zu trackende Bereich, Region Of Interest (ROI) genannt, eingeschränkt (Abbildung 32).

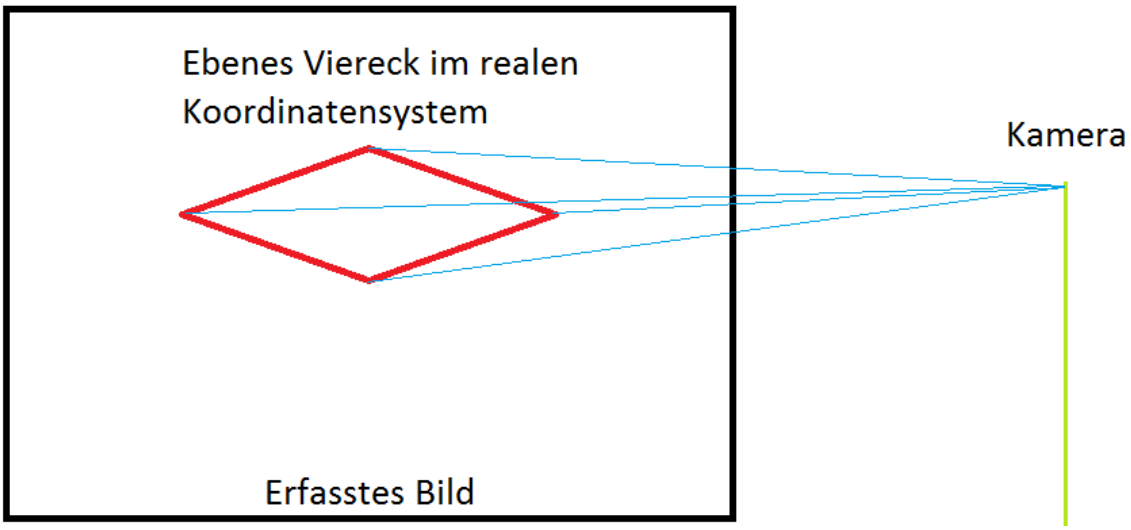

Abbildung 31: Links im schwarzen Rechteck ist das von der Kamera (rechts positioniert) erfasste Bild zu sehen. Das rote Viereck stellt ein durch die Kameraorientierung verzerrtes, ebenes Viereck dar.

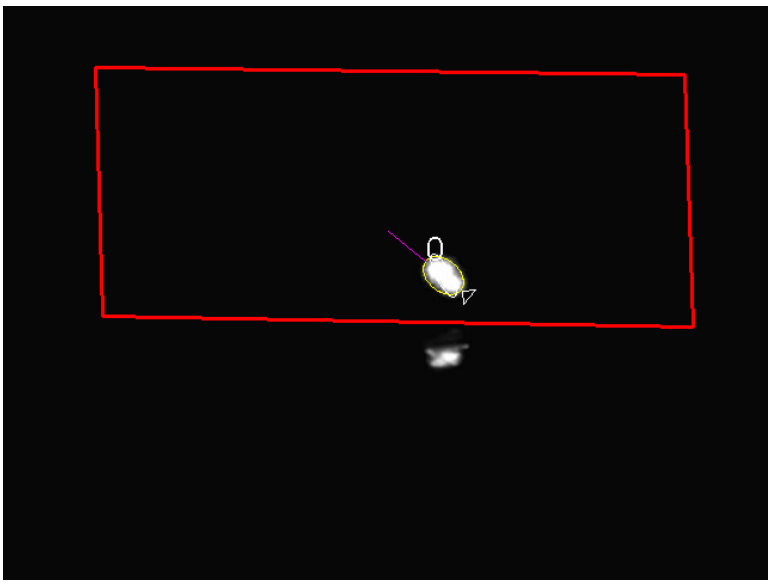

Abbildung 32: Der rote Bereich stellt den rektifizierten Bereich dar. Dieser Bereich dient als Region Of Interest, alle anderen Objekte außerhalb des Bereiches werden ignoriert.

Da das Aquarium aus Glas besteht und Glas Lichtwellen reflektiert (Abbildung 33), ist die Festlegung eines ROIs notwendig um Reflektionen während des Trackings ignorieren zu können. Der ROI wird durch den rektifizierten Bereich definiert.

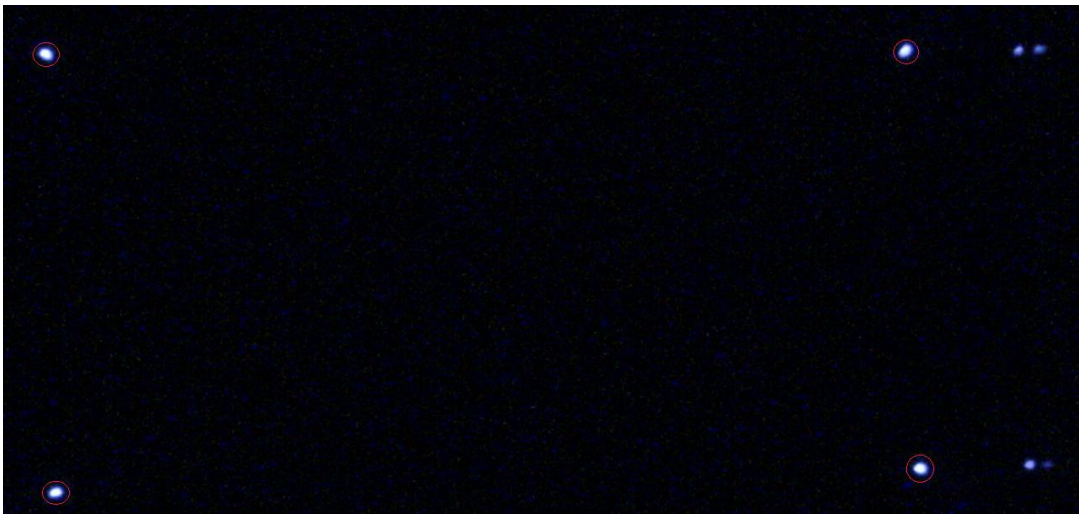

Abbildung 33: Ein erfasstes Kamerabild. Die rot eingekreisten Punkte zeigen die Ecken des Aquariums. Am rechten Bildrand sind Reflexionen zu erkennen, die vom Tracking nicht erfasst werden dürfen.

Um den Bereich korrekt zu rektifizieren, werden vier Infrarotlicht-LEDs an den Ecken des Aquariums befestigt (Abbildung 33). Der Benutzer muss vor Initialisierung des Trackings jede dieser vier LEDs einmal mit dem Mauszeiger (siehe Abschnitt 3.4) anklicken. Dadurch werden die Eckpunkte des Quadrates definiert (Abbildung 32) und anschließend alle Koordinaten rektifiziert. Wird währenddessen die ESC-Taste gedrückt, so wird der gesamte Bildbereich erfasst (Abbildung 40). Alle durch das Tracking erfassten Objekte müssen sich in diesem Bereich befinden und erhalten Positionswerte im Bereich zwischen 0 und 1. Das hat einen zusätzlichen Vorteil für den Benutzer der Klasse. Er kann immer davon ausgehen, dass sich die Positionen der gefundenen Objekte im selben Bereich zwischen (0,0) und (1,1) befinden und muss sich nicht mehr um die Kameraauflösung, die den Bildbereich festlegt (Abbildung 25), kümmern.

3.4 Tracking-Verfahren

Im Laufe der Entwicklung dieser Klasse wurden zwei Tracking-Verfahren entworfen und getestet. Der erste Ansatz wird zur Lokalisierung des ersten Prototyps (siehe Abschnitt 2.1.1) benutzt und verwendet pro Fisch-Replikat eine Infrarotlicht-LED. Der zweite Ansatz wird zur Lokalisierung des zweiten Prototyps (siehe Abschnitt 2.1.2) benutzt und verwendet pro Fisch-Replikat zwei Infrarotlicht-LEDs. Im Folgenden werden beide Ansätze vorgestellt, dabei werden die Probleme des ersten Ansatzes erläutert und gezeigt, weshalb ein zweiter Ansatz notwendig war.

3.4.1 Lokalisierung der LEDs im binarisierten Bild

In beiden in Abschnitt 3.4.2 und Abschnitt 3.4.3 entwickelten Ansätzen werden in jedem Frame zuerst die LEDs lokalisiert. Die durchgeführte Binarisierung (siehe Abbildung 34) liefert eine Matrix mit Nullen und Einsen, die das Bild repräsentiert. Durch

den festgelegten Schwellenwert ist wahrscheinlich, dass Einsen Pixel von leuchtenden LEDs sind.

0	0	0	0	1	1	1	0	0	0	0	1	1
0	0	0	1	0	1	1	1	1	1	0	0	1
0	0	0	0	0	1	1	1	1	1	0	1	1
0	0	0	0	0	1	1	1	1	1	0	1	0
0	0	0	0	0	0	1	1	1	0	0	0	1
0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

Abbildung 34: Ein Beispiel einer durch die in der Binarisierung (siehe Abschnitt 3.2.2) erzeugten Matrix. Die Einsen stellen die Bereiche des Bildes dar, in denen die Helligkeit der Pixel den festgelegten Schwellenwert überschreitet.

Es muss festgestellt werden, welche dieser Einsen zu welcher LED gehören. Hierzu werden Zusammenhangskomponenten in dieser Matrix gesucht und dadurch Pixel entsprechend zugeordnet. Wann sind zwei Einsen zusammenhängend?

Da die Matrix ursprünglich Pixel in einem zweidimensionalen Bild darstellt, muss betrachtet werden, auf welche Art und Weise Pixel miteinander zusammenhängend (benachbart) sein können. Es kommen zwei Möglichkeiten in Betracht, die 4-Konnektivität und die 8-Konnektivität [9]. 4-Konnektivität besagt, dass ein Pixel genau dann mit einem anderen Pixel benachbart ist, wenn sich diese eine Kante teilen [9]. 8-Konnektivität besagt, dass ein Pixel genau dann mit einem anderen Pixel benachbart ist, wenn sich diese eine Kante oder eine Ecke teilen [9].

Zwei benachbarte Pixel gehören eindeutig zu einer Zusammenhangskomponente. Ist zusätzlich ein *Pixel **A*** über einen anderen *Pixel **B*** mit *Pixel **C*** benachbart, so gehören *Pixel **A*** und *Pixel **C*** auch zur selben Zusammenhangskomponente.

Der Fall, dass nur diagonal benachbarte Pixel als einzelne Zusammenhangskomponenten betrachtet werden (Abbildung 35) muss ausgeschlossen werden, da diese einzelnen Pixel offensichtlich zu einer LED gehören. Daher wird bei der Zusammenhangskomponentensuche 8-Konnektivität verwendet.

0	0	0	0	1	1	1	0	0	0	0	1	1
0	0	0	1	0	1	1	1	1	1	0	0	1
0	0	0	0	0	1	1	1	1	1	0	1	1
0	0	0	0	0	1	1	1	1	1	0	1	0
0	0	0	0	0	0	1	1	1	0	0	0	1
0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

Abbildung 35: Die Matrix aus Abbildung 34. Es sind dunkelrot-markierte Pixel zu sehen, die offensichtlich zu einer LED gehören, mit 4-Konnektivität jedoch als einzelne LEDs erkannt werden würden.

Abbildung 34 zeigt eine Matrix mit drei Zusammenhangskomponenten. Der Computer kommt folgendermaßen zu diesem Ergebnis: Ein Algorithmus geht die Matrix hierfür zeilenweise durch und überprüft für jede Zelle, ob in dieser eine 1 steht. Ist dies der Fall, dann werden nach 8-Konnektivität entsprechende Nachbarn besucht und es wird geschaut, ob dort auch eine 1 steht (positive Nachbarn). Ist dem so, wird für jeden positiven Nachbarn gefragt, welcher Zusammenhangskomponente er angehört. Gehört kein Nachbar einer an, so entsteht aus der aktuellen Zelle und aller gefundenen positiven Nachbarn eine Zusammenhangskomponente. Sonst wird die aktuelle Zelle in alle Zusammenhangskomponenten aller positiven Nachbarn, die unterschiedlich sein können, eingefügt.

Beispiel: Hat eine *Zelle* ***X*** den positiven *Nachbarn* ***Y*** mit *Zusammenhangskomponente 1* und den positiven *Nachbarn* ***Z*** mit *Zusammenhangskomponente 2*, so wird ***X*** in *Zusammenhangskomponente 1* und *Zusammenhangskomponente 2* hinzugefügt und alle Zellen aus *Zusammenhangskomponente 1* werden in *Zusammenhangskomponente 2* hinzugefügt und umgekehrt. Beide Zusammenhangskomponenten haben jetzt dieselben Zellen und sind identisch. Wurde die Matrix fertig durchsucht, müssen identische Zusammenhangskomponenten zu einer zusammengefasst werden.

Dieser Algorithmus wird definiert als *findBlobsForRobofish()*. Wird er auf die Matrix aus Abbildung 34 ausgeführt, so stehen im Ergebnis drei Zusammenhangskomponenten, die in Abbildung 36 farblich eingezeichnet sind.

0	0	0	0	1	1	1	0	0	0	0	1	1
0	0	0	1	0	1	1	1	1	1	0	0	1
0	0	0	0	0	1	1	1	1	1	0	1	1
0	0	0	0	0	1	1	1	1	1	0	1	0
0	0	0	0	0	0	1	1	1	0	0	0	1
0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

Abbildung 36: Die durch unterschiedliche Farben hervorgehobenen Zusammenhangskomponenten nach Anwendung von *findBlobsForRobofish()* auf die Matrix aus Abbildung 34.

Da jede gefundene Zusammenhangskomponente die Pixel einer LED repräsentiert, kann ein Viereck (Blob) um diese Pixel gezogen.

Der Mittelpunkt des berechneten Blobs wird zur Position der durch die Zusammenhangskomponente repräsentierten LED (Abbildung 37).

Es kam in sehr seltenen Fällen vor, dass an einer Position, an der sich eindeutig keine LED befand, in nur einem Frame ein einzelnes Pixel als eigene Zusammenhangskomponente erfasst wurde. Hier handelte es sich eindeutig um Messfehler, da eine Infrarotlicht-LED nie durch nur einen Pixel repräsentiert werden kann. Dieser Fehler wurde behoben, in dem der Algorithmus mit einer zusätzlichen Abfrage, die überprüft, wie groß Zusammenhangskomponenten sein müssen, erweitert wurde. Ist dieser Wert größer als ein Schwellenwert den der Nutzer bei Bedarf festlegen kann, so handelt es sich um eine Zusammenhangskomponente. Ist das nicht der Fall, wird davon ausgegangen, dass es sich um einen Messfehler handelt und die Komponente wird verworfen (Abbildung 38).

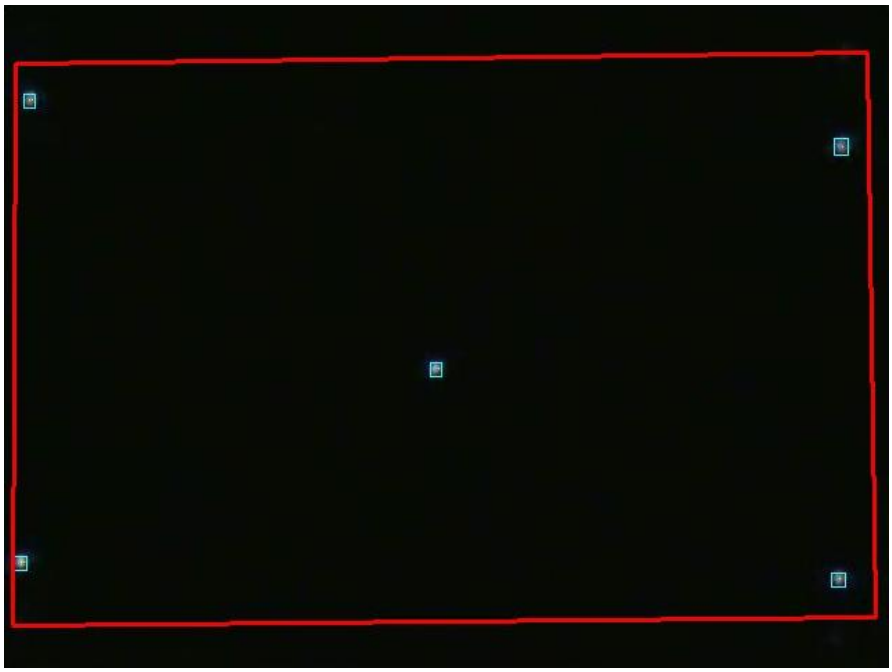

Abbildung 37: Gefundene LEDs (Zusammenhangskomponenten) im rektifizierten Bereich.

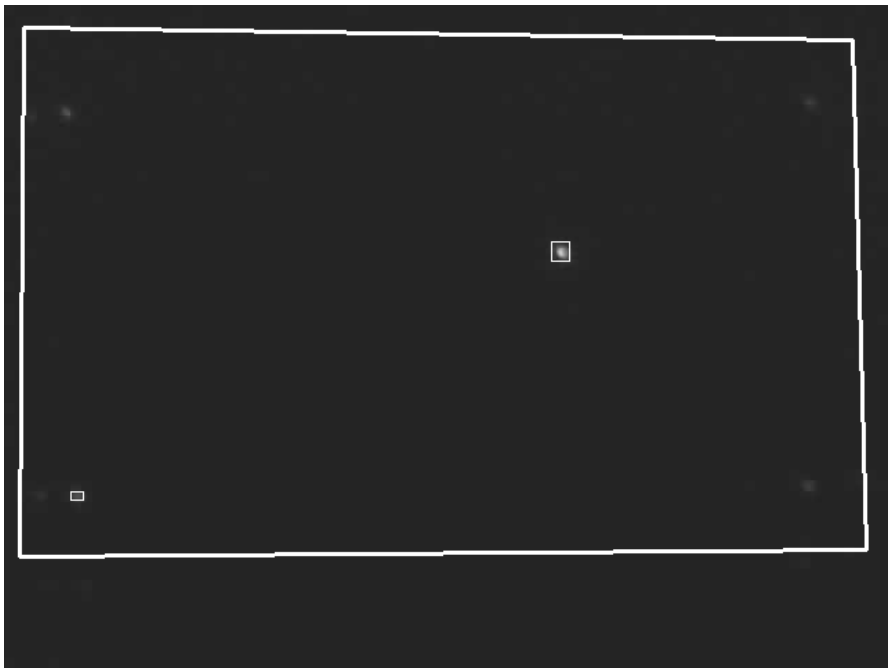

Abbildung 38: Dasselbe Bild aus Abbildung 37. Es werden weniger Zusammenhangskomponenten gefunden, da der Schwellenwert für die Mindestgröße einer Zusammenhangskomponente deutlich erhöht wurde. Kleinere Zusammenhangskomponenten (im Bild zu sehen links oben, rechts oben und rechts unten) werden ignoriert und nicht als LED erkannt.

3.4.2 Ansatz 1: Tracking mit einer Infrarotlicht-LED

Im ersten entwickelten Ansatz wurde davon ausgegangen, dass keine Richtungsbestimmung für die Positionskorrektur des Roboters (siehe Abschnitt 1.2.1) notwendig ist. Er verwendet Fisch-Replikate vom Prototyp 1 (siehe Abschnitt 2.1.1) mit einer integrierten

Infrarotlicht-LED. Die Fische werden in diesem Ansatz wie in Abschnitt 2.1.1 beschrieben vom Roboter gesteuert, jede LED repräsentiert dabei einen Roboterfisch.

Alle gefundenen Roboterfische werden in einer dynamischen Liste gespeichert. Es können beliebig viele lokalisiert werden. Ein Roboterfisch wird in einer eigenen Datenstruktur gespeichert. Die Datenstruktur enthält die aktuelle Position des Objektes, eine eindeutige Identifikationsnummer, die für jeden neuen Roboterfisch inkrementiert wird, eine Historie, die eine festgelegte Anzahl an vorherigen Positionen speichert und eine Time To Live (TTL). Die TTL definiert, wie viele Frames ein Roboterfisch vom Tracking nicht lokalisiert werden kann bevor er als verloren gilt. Sie wird in jedem Frame reduziert, falls dem aktuellen Objekt keine LED zugeordnet werden kann. Wurde eine Zuordnung gefunden, so wird die TTL wieder auf den Maximalwert, der vom Benutzer einstellbar ist, erhöht. Wenn die TTL den Wert 0 erreicht, dann wird der Roboterfisch als *nicht aktiv* (tot) erfasst und vom Tracking nicht mehr behandelt. Aus Testzwecken wird er nicht aus der Roboterfisch-Liste gelöscht, sondern bleibt mit den zuletzt gesichteten Koordinaten gespeichert. Das soll verdeutlichen, dass das Tracking Objekte verloren hat. Um einen Roboterfisch eindeutig einer Zusammenhangskomponente, also einer LED (siehe Abschnitt 3.4.1), zuordnen zu können, wird ein neuer Schwellenwert, *maxFishMovement*, benutzt. Dieser Wert legt fest, wie viel Pixel sich ein Roboterfisch innerhalb von zwei aufeinanderfolgenden Frames bewegen darf, um noch als gleiches Objekt erkannt zu werden. Da ein Fisch-Replikat von einem Roboter mit einem Durchmesser von 17 cm (siehe Abschnitt 2.2) gelenkt wird und das Replikat einen maximalen Durchmesser von 7 cm (siehe Abschnitt 2.1) hat, haben zwei Roboterfische immer einen Mindestabstand von 17 cm zueinander. Abhängig von der Kamerahöhe und der verwendeten Auflösung muss der Benutzer der Klasse den *maxFishMovement* Wert so wählen, dass er groß genug ist damit Roboterfische, die sich mit maximaler Geschwindigkeit bewegen, noch als identische Objekte erkannt werden. Der Wert muss zudem klein genug sein, so dass zwei verschiedene Roboterfische nie verwechselt werden können. Der Benutzer hat auch die Option, sich diesen Wert automatisiert von der Klasse berechnen zu lassen. Hierfür wird der Durchmesser einer beliebigen Zusammenhangskomponente verwendet und mit einem Faktor multipliziert. Mit Hilfe dieses Ansatzes können Zusammenhangskomponenten eindeutig Roboterfischen zugeordnet (siehe Anhang 1: Pseudocode für Ansatz 1) werden.

Wird der obige Algorithmus beispielweise mit einer minimalen Komponentenzusammenhangsgröße von fünf Pixeln und einem Binarisierungs-Schwellenwert von 50 auf das statische Bild aus Abbildung 37 angewendet, so werden fünf Fisch-Replikate gefunden (Abbildung 39).

```
Printing all fishes:
-----------------------------------------------------------------
Fish Nr.: [0] --- Position: (0.910937,0.14375) --- Position (Image): (583,69) --- TTL: 9 --- ID: 0 ---
-----------------------------------------------------------------
-----------------------------------------------------------------
Fish Nr.: [1] --- Position: (0.0703125,0.160417) --- Position (Image): (45,77) --- TTL: 9 --- ID: 1 ---
-----------------------------------------------------------------
-----------------------------------------------------------------
Fish Nr.: [2] --- Position: (0.629687,0.370833) --- Position (Image): (403,178) --- TTL: 9 --- ID: 2 ---
-----------------------------------------------------------------
-----------------------------------------------------------------
Fish Nr.: [3] --- Position: (0.907812,0.722917) --- Position (Image): (581,347) --- TTL: 9 --- ID: 3 ---
-----------------------------------------------------------------
-----------------------------------------------------------------
Fish Nr.: [4] --- Position: (0.0828125,0.739583) --- Position (Image): (53,355) --- TTL: 9 --- ID: 4 ---
-----------------------------------------------------------------
```

Abbildung 39: Fünf gefundene Fisch-Replikate nach Anwendung von Ansatz 1. Der erste Wert zeigt die Position des Replikates in der Liste *my_fishes*, der zweite Wert stellt die Position im rektifizierten Bereich dar, der dritte Wert die Position im Bild (Pixelkoordinaten), der vierte Wert zeigt die TTL und der letzte Wert zeigt die erhaltene ID. Binarisierungs-Schwellenwert: 50. Minimale Komponentenzusammenhangsgröße: 5 Pixel.

Abbildung 40 Visualisiert den oben beschriebenen Ansatz: Das Fisch-Replikat befindet sich mittig im Bild und bewegt sich in eine Richtung. Die gelben Punkte zeigen die gefahrene Route. Es wurden dieselben Schwellenwerte wie in Abbildung 39 benutzt. Der *maxFishMovement* Wert wurde auf 10 Pixel gesetzt. Bei einer Auflösung von 640x480 Pixel wurde mit der in Abschnitt 3.1 verwendeten Entwicklungsumgebung und der Kamera aus Abschnitt 2.3 eine Framerate von 25 Frames pro Sekunde festgestellt.

Dieser Ansatz hat sich als unbrauchbar herausgestellt. Nach wenigen Testversuchen in Zusammenhang mit dem von Hai Nguyen entworfenen Regler (siehe Abschnitt 1.2.1), wurde deutlich, dass die Feststellung der Orientierung des Roboters für eine genaue Positionskorrektur in Kombination mit dem Regler wahrscheinlich unabdingbar ist. Außerdem war es nicht notwendig, das System dynamisch zu halten, da in einem Experiment von Anfang an klar ist, wie viele Fisch-Replikate (und damit Roboter) verwendet werden. Zusätzlich liefert dieser Ansatz dem Nutzer keine Möglichkeit einer Ausnahmebehandlung, falls ein Fisch-Replikat im Tracking verloren geht. Im schlimmsten Fall wird die TTL des verlorenen Fisches so lange dekrementiert, bis sie den Wert 0 erreicht. Dann wird der Fisch als inaktiv markiert. Würde er wieder einen Frame später erfasst werden, so würde der Algorithmus diesen als neuen Fisch erkennen und ihm eine neue ID zuweisen. Der Nutzer verliert die Steuerung über den verloren gegangenen Roboter, da dessen Koordinaten nicht mehr aktualisiert werden.

Deshalb wurde ein neuer Ansatz basierend darauf, dass in einem Fisch-Replikat zwei Infrarotlicht-LEDs verbaut werden (siehe Abschnitt 2.1.2), entworfen und getestet.

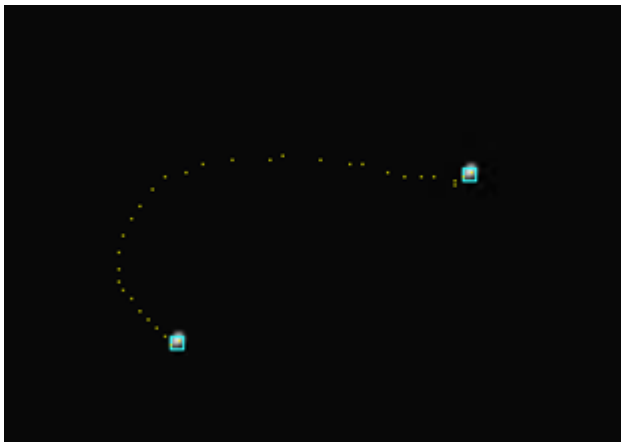

Abbildung 40: Ansatz 1 visualisiert. Ein Fisch-Replikat wird vom Algorithmus erfasst. Die untere Zusammenhangskomponente stellt die Position zu Anfang des Experiments dar, die obere Zusammenhangskomponente die Endposition. Die gelben Punkte zeigen die gefahrene Trajektorie.

3.4.3 Ansatz 2: Tracking mit zwei Infrarotlicht-LEDs

Wie in Abschnitt 3.4.2 erläutert, ist die Feststellung der Orientierung des Roboters für eine genaue Positionskorrektur und damit für dessen Steuerung im implementierten Regler notwendig. Um eine Orientierung im zweidimensionalen Raum eindeutig feststellen zu können, sind mindestens drei Punkte notwendig (Abbildung 41).

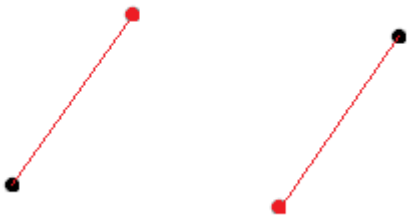

Abbildung 41: Jeweils zwei verbundene Punkte im zweidimensionalen Koordinatensystem. Die Orientierung ist unklar, da jeweils beide Endpunkte beider Geraden sowohl als Vorderseite als auch als Hinterseite des Roboters definiert werden können. Eine Startannahme ist daher notwendig.

Drei LEDs passen nicht in ein Fisch-Replikat (siehe Abschnitt 2.1), weswegen zwei benutzt werden und mit einer Annahme gearbeitet wird: Es wird angenommen, dass die Infrarotlicht-LED mit der kleineren x-Koordinate die LED an der Vorderseite des Fisch-Replikates ist. Wie in Abschnitt 2.1.2 beschrieben, soll das Replikat so positioniert werden, dass die durch zwei Infrarotlicht-LEDs definierte Achse die Achse des Roboters darstellt. War die Annahme richtig, so müsste sich, wenn der Roboter geradeaus fährt, die Achse in dieselbe Richtung wie die des Roboters, also nach vorne bewegen. Ist dies nicht der Fall (bewegt sie sich nach hinten), so müssen beide Punkte vertauscht werden (der vordere Punkt wird zum hinteren Punkt und umgekehrt). Diese Vertauschung steht dem Benutzer innerhalb der Klasse in einer *swap()* Methode zur Verfügung. Alternativ

kann er den Roboter vor Initialisierung des Trackings auch so positionieren, dass dieser in Bezug zur vertikalen Bildachse in einem Winkel zwischen 90° und 270° steht (Abbildung 42). Damit wird sichergestellt, dass die Infrarotlicht-LED an der Vorderseite des Roboterfisches in jedem Fall einen kleineren x-Koordinatenwert hat.

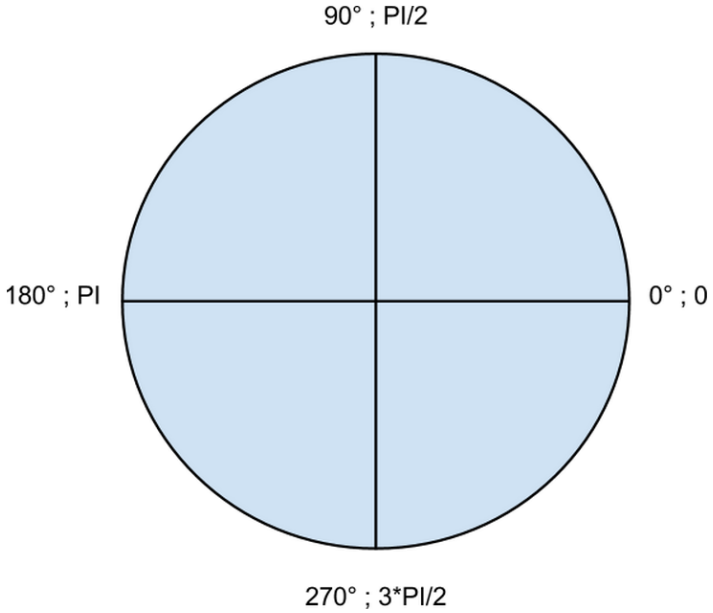

Abbildung 42: Die verwendete Orientierung im Bild. An den Seiten sind jeweils die Winkel im Gradmaß und im Bogenmaß angegeben.

In zweiten Ansatz werden zwei LEDs einem Fisch-Replikat zugeordnet und anschließend wird die Orientierung berechnet. Als Position wird der Mittelpunkt zwischen beiden LEDs festgelegt. Die Fischstruktur aus Ansatz 1 wird um zwei neue Punkte (vordere LED, hintere LED) und einen Winkel erweitert. Um herauszufinden, ob zwei LEDs einem Roboterfisch gehören, wird ein Schwellenwert (*maxLEDdistance*) benutzt. Dieser gibt an, wie groß die maximale Distanz zwischen zwei Infrarotlicht-LEDs in einem Replikat sein darf. Diese ist, ähnlich wie *maxFishMovement* im ersten Ansatz (siehe Abschnitt 3.4.2), von der Kamerahöhe abhängig. Der Benutzer kann diesen Wert entweder automatisiert von der Klasse berechnen lassen oder selber festlegen. Für die automatisierte Berechnung wird die minimale Distanz zweier Zusammenhangskomponenten, also zweier LEDs, verwendet und mit einem Faktor multipliziert. Auch hier existiert ähnlich wie bei *maxFishMovement* eine durch den Durchmesser des Roboters (17 cm) bedingte minimale Distanz zwischen zwei Roboterfischen, nämlich 17 cm. Die beiden Infrarotlicht-LEDs sind in einem Abstand von 2 cm verbaut worden (siehe Abschnitt 2.1.2), somit bietet der Aufbau dem Benutzer bei der Festlegung vom *maxLEDdistance*-Schwellenwert einen gewissen Spielraum. Mit Hilfe von *maxLEDdistance* und einer Funktion *findLEDpairs()* werden zwei Zusammenhangskomponenten zu einer neuen LED_PAIR-Zusammenhangskomponente zugeordnet und in einer dynamischen Liste *currentPairs* gespeichert (siehe Anhang 2: Pseudocode für *findLedPairs()*).

Im zweiten Ansatz wird davon ausgegangen, dass die Anzahl der Fische bekannt ist. Damit der Nutzer der Klasse nicht jedes Mal angeben muss, wie viel Fisch-Replikate und Roboter er verwenden will, wird die Anzahl automatisiert innerhalb einer Initialisierungsphase festgestellt. Innerhalb dieser wird erwartet, dass alle Roboter entsprechend im Trackingbereich positioniert wurden. Der Benutzer kann nun beliebig die Schwellenwerte so lange verändern, bis alle aufgestellten Fisch-Replikate vom Tracking erfasst wurden. Um zu vermeiden, dass Roboterfische durch nachträgliche Änderungen der Schwellenwerte verloren gehen oder hinzukommen, dürfen diese Werte nach der Initialisierung nicht mehr verändert werden. Dabei wird in einer Schleife die Funktion *findFirstFishes()* ausgeführt, bis der Benutzer mit dem Ergebnis zufrieden ist und alle Objekte gefunden wurden. Diese Funktion greift ein neues Kamerabild ab, sucht sich mit Hilfe der in Abschnitt 3.4.1 beschriebenen Binarisierung die Zusammenhangskomponenten, ruft anschließend *findLEDPairs()* auf und fügt einen neuen Fisch für jedes Element aus *currentPairs* in die Liste *my_fishes* ein (siehe Anhang 3: Pseudocode für *findFirstFishes()*).

Ist die Initialisierung beendet worden kann das Tracking, nämlich die permanente Zuordnung von LED-Paaren zu Roboterfischen und die Orientierungsbestimmung, beginnen. Hierfür wird die in Ansatz 2 neu definierte Funktion *findRobofishes()* (siehe Anhang 4: Pseudocode für Ansatz 2) periodisch aufgerufen. *maxFishMovement* wurde entfernt. Es wird stattdessen in jedem neuen Bild überprüft, welches LED-Paar den geringsten Abstand zu einem zuvor erfassten Roboterfisch hat und diesem zugeordnet. Da nun bekannt ist, wie viele Roboterfische gesucht werden, sollten im Idealfall immer entsprechend viele LED-Paare gefunden werden. Es wird davon ausgegangen, dass zwischen zwei Frames der Roboter keine unrealistisch großen Distanzen (Sprünge) zurücklegt und dadurch zwei Roboter ihre Position nicht tauschen können. Dadurch ist eine eindeutige Zuordnung möglich. Verliert das Tracking einen Roboterfisch für TTL Frames, so wird (wie in Ansatz 1) der Roboterfisch nicht aus der Liste gelöscht und dessen *active* Attribut auf *false* gesetzt. Im Gegensatz zu Ansatz 1 kann hier nach Verlust ein Roboterfisch eindeutig wiedergefunden werden. Im ungünstigsten Fall, wenn z.B. die Kamera für mehrere Sekunden verdeckt wurde, hat das Tracking alle Objekte verloren. Hier kann nur mit einer Wahrscheinlichkeit die Aussage getroffen werden, welches Objekt zu welchem Roboter gehört (z.B. können zwei Roboter genau in diesem Zeitintervall ihre Position getauscht haben und somit verwechselt werden). Da die Kamera jedoch beim Experimentieren nicht verdeckt werden sollte, ist das Eintreffen des ungünstigsten Falles sehr unwahrscheinlich und kann ignoriert werden. Der Vorteil dieses Ansatzes ist, dass der Nutzer der Klasse Ausnahmebehandlungen für verlorene Objekte treffen kann, indem er das *active* Attribut periodisch abfragt und im Falle, dass es *false*

ist, z.B. den Roboter so lange die Trajektorie zurück fahren lässt, bis er wieder vom Tracking erfasst wurde.

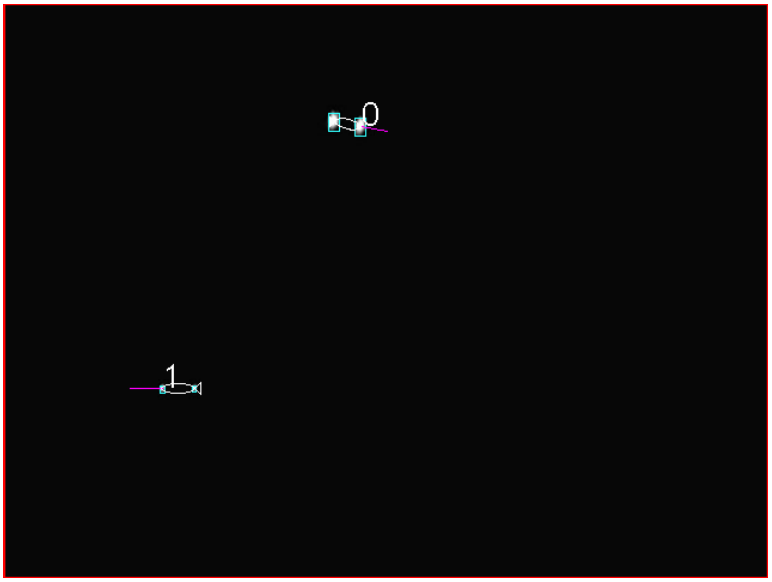

Abbildung 43: Visualisierung von findRobofishes() aus Ansatz 2. Der Algorithmus hat zwei Fisch-Replikate im Bild lokalisiert. Der obige hat die ID 0, der untere hat die ID 1. Die dunkelrote Linie stellt die Orientierung des Roboters dar.

Abbildung 43 zeigt die Anwendung vom in Ansatz 2 beschriebenen Algorithmus auf zwei Fisch-Replikate. Am oberen Roboterfisch (ID 0) ist zu erkennen, dass die beiden von der Kamera erfassten Infrarotlicht-LEDs heller leuchten als die vom unteren Roboterfisch (ID 1). Die Infrarotlicht-LEDs strahlen ihr Licht vertikal nach oben. Je mittiger sich die Infrarotlicht-LEDs im Bild befinden, desto heller leuchten sie. Je näher sie sich dem Rand nähern, desto schwacher leuchten sie. Das liegt daran, dass die LEDs einen kleinen Abstrahlwinkel besitzen und dadurch in bestimmten Positionen weniger Lichtwellen auf die Linse treffen. Es kann passieren, dass die in der Initialisierung festgelegten Schwellenwerte an einigen Positionen im Bild andere Ergebnisse erzielen und dadurch ein Roboterfisch eventuell nicht mehr korrekt lokalisiert werden kann.

Dieser Fall lässt sich umgehen. Wird die die Belichtungszeit der Kamera erhöht, so verschmelzen beide LEDs zu einer Zusammenhangskomponente (Abbildung 44).

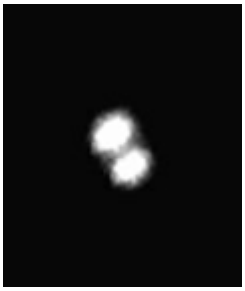

Abbildung 44: Durch Erhöhung der Belichtungszeit der Kamera verschmelzen zwei LEDs zu einer.

Wird eine Ellipse um die Zusammenhangskomponente gezogen, so ist eine Seite länger als die andere. Diese Information kann nun verwendet werden, um auf die einzelnen Positionen beider verschmolzener Infrarotlicht-LEDs zurück zu schließen. Die beiden Punkte, die am weitesten voneinander entfernt sind, werden als Mittelpunkte der beiden LEDs verwendet (Abbildung 45).

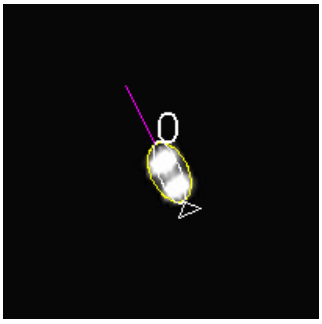

Abbildung 45: Aus einer Zusammenhangskomponente (gelb eingezeichnete Ellipse) konnten die Positionen beider LEDs bestimmt, der Roboterfisch lokalisiert und seine Orientierung berechnet (dunkelrote Linie) werden.

Für diese optionale Lösung wurde die Klasse um eine boolesche Variable *expectTwoBlobs* erweitert. Diese gibt an, ob die durch *findBlobsForRobofish()* gefundenen Zusammenhangskomponenten als einzelne Infrarotlicht-LEDs oder zwei verschmolzene Infrarotlicht-LEDs behandelt werden sollen.

Mit Ansatz 2 konnte bei der verwendeten Kamera (siehe Abschnitt 2.3) und der in Abschnitt 3.1 benutzten Entwicklungsumgebung bei einer Auflösung von 640x480 Pixeln im Durchschnitt eine Framerate von 23 Frames pro Sekunde festgestellt werden.

3.5 Beispielprogramm

Ein Beispielprogramm wurde mit der QT Library entwickelt, um die Klasse effizient testen zu können und um eine robuste Beispielimplementierung innerhalb einer GUI darzustellen (Abbildung 46).

Das Beispielprogramm zeigt, wie alle Methoden der Klasse robust und anwenderfreundlich implementiert werden können. Außerdem lassen sich mit dem Programm eine Vielzahl an Szenarien simulieren und ausprobieren (mehr dazu in Abschnitt 3.5.1). Das Programm ist in mehrere Abschnitte eingeteilt. Im „Threading“-Abschnitt lässt sich eine Tracking-Instanz starten und beenden. „Print Fishes“ gibt in einer Konsole die gefundenen Roboterfische mit allen verfügbaren Informationen aus (Abbildung 39). Zwei „Toggle“ Schalter steuern die Einstellung für die automatisierte Berechnung von maxLED-distance und die Einstellung, ob eine Zusammenhangskomponente eine LED oder zwei LEDs darstellt. Mittig lassen sich die Schwellenwerte steuern. Die Rektifizierung lässt sich bei Bedarf einleiten und wiederholen. Unter „See what happens“ befinden sich verschiedene Testszenarien (siehe Abschnitt 3.5.1 bis 3.5.3).

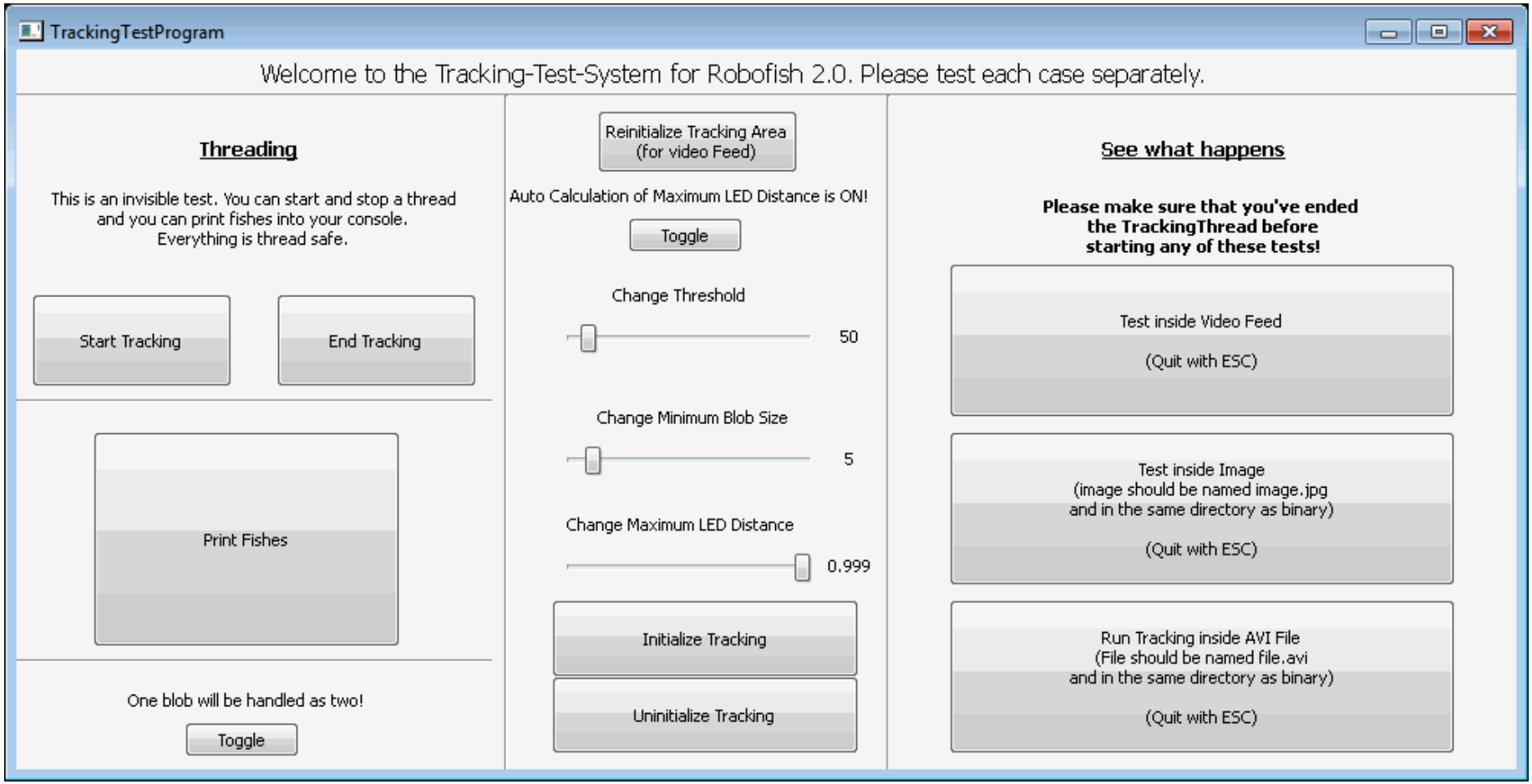

Abbildung 46: Das entworfene Beispielprogramm. Links lässt sich eine Tracking-Instanz starten und stoppen. Mit „Print Fishes“ werden die gefundenen Roboterfische in einer Konsole ausgegeben. Links unten ist eine Schaltfläche, mit der die Variable *expectTwoBlobs* verändert werden kann. Mittig lassen sich Schwellenwerte festlegen, das Tracking initialisieren, die Initialisierung wieder aufheben, die Rektifizierung wiederholen und einstellen, ob die maximale Distanz zwischen zwei Infrarotlicht-LEDs automatisiert berechnet werden soll. Rechts befinden sich verschiedene visualisierte Testszenarien.

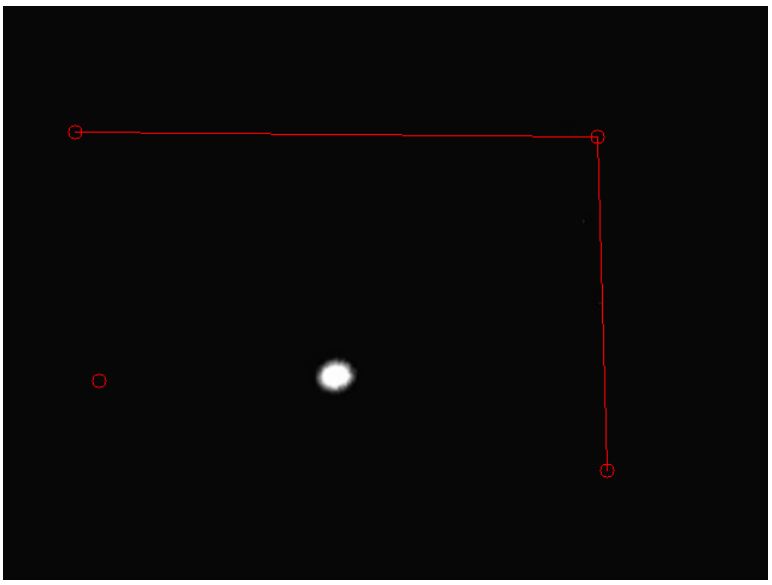

Abbildung 47: Der rektifizierte Bereich wird anhand von vier Punkten, die vom Benutzer festgelegt werden, definiert. Dabei sieht der Benutzer sofort den Bereich im Bild und kann ihn bei Bedarf korrigieren. Mit der ESC-Taste lässt sich die Rektifizierung automatisieren. Dann wird der gesamte Bildbereich verwendet.

Das Programm folgt dabei der in Abschnitt 3.4.3 beschriebenen Logik: Solange das Tracking nicht initialisiert wurde, lässt sich das Tracking nicht starten, noch lassen sich die Testszenarien ausführen. Vor der Initialisierung sind nur die beiden „Toggle“ Buttons aktiv und haben dadurch eine Wirkung. Alle Regler lassen sich verändern. Zuerst muss die ROI festgelegt und rektifiziert werden. Ist dies noch nicht geschehen, leitet die Programmlogik die Rektifizierung vor Initialisierungsbeginn automatisch ein (Abbildung 47). Während der Initialisierung lassen sich die Schwellenwerte beliebig verändern, die „Toggle“ Schalter sind jedoch deaktiviert. Die Initialisierung ruft periodisch *findFirstFishes()* auf und zeigt ein Fenster mit allen gefundenen Objekten an. Werden die Schwellenwerte geändert, so wird der Einfluss jedes Schwellenwertes auf das Tracking sofort sichtbar und dadurch dem Benutzer verständlich erklärt. Ist das Ergebnis zufrieden stellend, kann mit der ESC-Taste die Initialisierung abgeschlossen werden. Danach werden alle Regler gesperrt und die beiden „Toggle“ Schalter bleiben weiterhin deaktiviert (Abbildung 48).

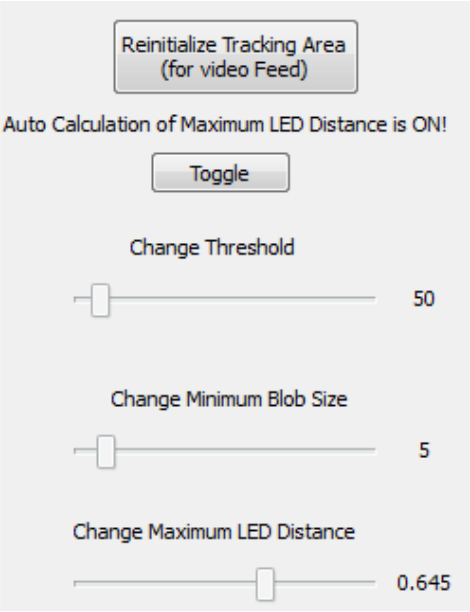

Abbildung 48: Die Initialisierung wurde abgeschlossen und die Regler sind eingefroren und dadurch deaktiviert. Der „Toggle“ Schalter für die automatisierte Berechnung von maxLEDdistance hat keine Wirkung mehr.

Jetzt erst lassen sich die verschiedenen Testszenarien starten. Wurde ein Szenario gestartet, so lässt sich kein anderes mehr einleiten, bis das aktuell ausgeführte beendet wurde. Wird ein Szenario beendet, so wird die Initialisierung automatisch aufgehoben. Alle Schwellenwerte lassen sich erneut einstellen und die „Toggle“ Schalter sind wieder aktiv. Bevor ein neues Szenario gestartet wird, muss das Tracking wieder initialisiert werden.

3.5.1 Kamerafeed-Testszenario

Im Kamerafeed-Testszenario wird kontinuierlich lokalisiert und visualisiert. Dem Benutzer wird ein Fenster angezeigt, in dem der Kamera-Feed inklusive gefundener Roboterfische sichtbar ist (Abbildung 43).

Mit Hilfe dieses Testszenarios lassen sich durch die Visualisierung die Schwellenwerte und physikalischen Parameter (Höhe der Kamera, Belichtungszeit, etc.) für den Anwender besser einstellen.

3.5.2 Bild-Testszenario

Da nicht immer eine Kamera und die nötige Hardware (Infrarotlicht-LEDs und Fisch-Replikate) zur Verfügung stehen, wurde das Bild-Testszenario erstellt. Es lokalisiert Objekte in einem statischen Bild. Für eine korrekte Lokalisierung sollte das Bild dem Kamerabild ähnlich sein und dadurch das Setup gut simulieren (Abbildung 49).

Abbildung 49: Ein Beispiel eines geeigneten Bildes für das Bild-Testszenario.

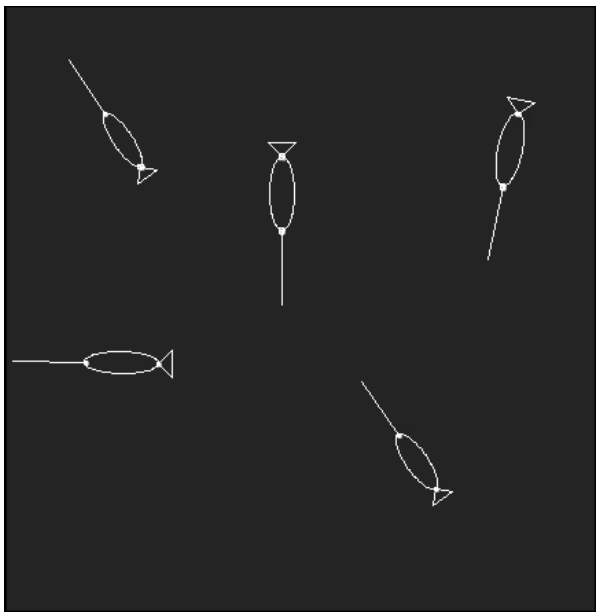

Abbildung 50: Die Ausführung des Bild-Testszenarios auf das in Abbildung 49 erstellte Bild. Es wurden fünf Roboterfische gefunden.

Da es sich um ein statisches Bild handelt, ist ein Tracking, also eine fortlaufende Lokalisierung und Orientierungsfindung, nicht notwendig. Ebenso wird keine Entzerrung durchgeführt, da von einem bereits entzerrten Bild ausgegangen wird.

Das Bild-Testszenario ruft einmalig *findFirstFishes()* auf, malt die gefundenen Objekte in das Bild und stellt es anschließend dem Nutzer innerhalb eines Fensters dar (Abbildung 50).

3.5.3 Videodatei-Testszenario

Um identische Szenarien testen zu können, wurde das Videodatei-Testszenario verwendet. Es gleicht dem Kamerafeed-Testszenario (siehe Abschnitt 3.5.1) mit dem Unterschied, dass eine Videodatei anstelle eines Kamerabildes für das Tracking verwendet wird. So ist es z.B. möglich, den Roboter eine bestimmte Trajektorie fahren zu lassen und mit der Kamera das Experiment aufzunehmen und abzuspeichern. Anschließend wird die gespeicherte Datei für das Testszenario verwendet. Durch die Modifikation der von der Klasse gestellten Variablen lassen sich verschiedene Konfigurationen auf ein identisches Experiment anwenden und den Einfluss der einzelnen Parameter direkt beobachten.

4 Validierung

Um die entworfenen Verfahren und die Logik der entwickelten Klasse ausgiebig testen zu können, wurden verschiedene Testszenarien in einem Testprogramm (siehe Abschnitt 3.5) implementiert und visualisiert. Dabei sollte ermittelt werden, in welchen Grenzen (Kamerahöhe und Schwellenwerte) die im Rahmen dieser Arbeit implementierten Verfahren korrekt funktionieren.

Die Kamerahöhe muss so gewählt werden, dass die Kamera mindestens den gesamten Bereich des Aquariums erfassen kann und dass durch Modifikation der Schwellenwerte eine eindeutige Lokalisierung des Fisch-Replikates in allen Bereichen des Aquariums möglich ist. Für einen 1x1 m Bereich ist eine Kamerahöhe von 1,30 m ausreichend, daher wurde vorwiegend mit dieser Kamerahöhe getestet. Die Variable *maxLED-distance* wird automatisiert vom System berechnet. Da die maximale Distanz zwischen zwei LED's und der Durchmesser des Roboters (siehe Abschnitt 2.1.2 und Abschnitt 2.2) nicht veränderbar sind, können wir diese Variable in der Validierung statisch wählen.

Für die Validierung der korrekten Lokalisierung wurden Roboterfische an verschiedenen Positionen in das Bild gestellt und es wurde getestet, mit welchen Schwellwerten diese vom Tracking-System gefunden wurden (siehe Tabelle 3 und Tabelle 4). Dabei stellte sich heraus, dass die optimale Belichtungszeit stark von der Variable *expectTwoBlobs* (siehe Abschnitt 3.4.3) abhängt. Ist diese auf *true* gesetzt, so wird eine geringe Belichtungszeit benötigt, ist sie auf *false* gesetzt, so muss die Belichtungszeit erhöht werden. Mit einer hohen Belichtungszeit wurden in einigen Szenarien mehr Roboterfische erkannt, als tatsächlich vorhanden waren (siehe Tabelle 3, letzte Zeile). Daher sollte der Binarisierungs-Schwellenwert entsprechend angepasst werden um falsche Erkennungen durch im Bild heller scheinende Objekte (aufgrund höherer Belichtungszeit) vorzubeugen (siehe Tabelle 3, vorletzte Zeile). Außerdem wurde festgestellt, dass der Binarisierungs-Schwellenwert und die minimale Komponentenzusammenhangsgröße voneinander abhängig sind. Je höher der Binarisierungs-Schwellenwert, desto kleiner werden die Zusammenhangskomponenten. Werden sie zu klein, dann werden sie vom Algorithmus verworfen. Um ein korrektes Tracking zu erreichen, muss der Benutzer während der Initialisierung des Trackings diese Schwellenwerte optimal einstellen. Für die Feststellung dieser Einstellungen wurden in der Validierung die in Abschnitte 3.5.1-3.5.3 erläuterten Testszenarien verwendet.

Um die Flexibilität des Systems zu zeigen wurden zusätzliche Tests mit verschiedenen Kamerahöhen gemacht (siehe Tabelle 4) und es wurde gezeigt, dass mit entsprechenden Schwellenwerten auch bei hoher Kameraposition eine Lokalisierung möglich ist.

Die vom System errechneten Positionen wurden durch den direkten Vergleich mit tatsächlichen Positionsmessungen, die manuell erfolgt sind, validiert (siehe Tabelle 1). Hierfür wurde an die Mitte des Roboters ein roter Punkt geklebt, der bei heller Belichtungszeit sehr gut zu erkennen war. Dann wurde (ohne aktives Tracking) dieser Punkt mit dem Mauszeiger angeklickt und es wurden die rektifizierten Koordinaten des Punktes errechnet. Die Belichtungszeit wurde anschließend wieder auf den entsprechenden Schwellenwert verringert und das Tracking wurde gestartet. Die vom Tracking errechneten Werte wurden mit den manuell gemessenen Werten verglichen (siehe Tabelle 1). Für die Validierung der Orientierung wurde an der Vorderseite des Roboters eine rote Stange geklebt, die bei heller Belichtungszeit sichtbar ist. Mit Hilfe dieser konnte die genaue Orientierung in Bezug auf das Bildkoordinatensystem errechnet werden (siehe Abbildung 51). Anschließend wurde das Tracking gestartet und die vorher manuell errechnete Orientierung mit der vom Tracking verglichen (siehe Tabelle 2).

Die Auswertung der Ergebnisse zeigt, dass die implementierten Algorithmen zur Lokalisierung und Orientierungsfindung der Roboterfische bei optimal gewählten Schwellenwerten korrekt funktionieren. Es ist zu erkennen, dass die zweite in Ansatz 2 entwickelte Lokalisierungsmethode (*expectTwoBlobs=false*, siehe Abschnitt 3.4.3) im Gegensatz zur ersten bei der Positionsbestimmung und Orientierungsfindung eine größere Ungenauigkeit aufweist.

Die Verwendbarkeit des Trackings zur Positionskorrektur des Roboters wurde mit Hilfe des von Hai Nguyen implementierten Reglers validiert. Dabei wurde eine Trajektorie im rektifizierten Bereich gezeichnet, die vom Roboter mit allen möglichen Geschwindigkeiten abgefahren werden sollte. Mit Hilfe der vom Tracking errechneten Position und Orientierung konnten Fahrfehler erfolgreich korrigiert und der Roboter dadurch an die korrekte Position navigiert werden (siehe Abbildung 52).

Abbildung 51: Im obigen Bild wurde an die Vorderseite des Roboters eine Stange befestigt, die die tatsächliche Orientierung des Roboters (270°, hier grün manuell eingezeichnet) zeigt. Im unteren Bild wurde die Orientierung durch das Tracking berechnet und eingezeichnet (ebenfalls 270°).

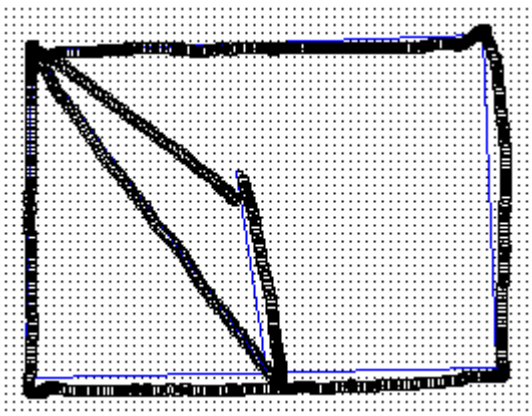

Abbildung 52: Die blauen Linien stellen die abzufahrende Trajektorie dar, die schwarzen Kreise zeigen die tatsächlich abgefahrene Trajektorie. Der Roboter startet in der Mitte, fährt zur oberen linken Ecke,

fährt dann ein Viereck und anschließend von der oberen linken Ecke zum Mittelpunkt der unteren horizontalen Linie des Vierecks. Zum Schluss fährt er in die Mitte und hält an. Die sichtbaren minimalen Abweichungen (z.B. an der rechten vertikalen Linie) zeigen noch Fahrfehler, die durch eine Optimierung des Reglers vermieden werden müssen.

Tatsächliche Koordinaten (x,y)	Errechnete Koordinaten (x,y)	expectTwoBlobs (true/false)	Belichtungszeit
(0.494313,0.440833)	(0.497752,0.441007)	*true*	5
(0.494313,0.440833)	(0.495313,0.433333)	*false*	*30*
(0.048602,0.028962)	(0.048437,0.029016)	*true*	*5*
(0.048602,0.028962)	(0.046562,0.026666)	*false*	*30*
(0.955603,0.0610211)	(0.955425,0.060916)	*true*	*5*
(0.955603,0.0610211)	(0.950201,0.065943)	*false*	*30*

Tabelle 1: Beispiel der eingesetzten Validierungsmethodik für die Position. *Kamerahöhe***: 1,30 m**. *Binarisierungs-Schwellenwert*: **50**. *Minimale Zusammenhangkomponentengröße*: **5**. *maxLEDdistance:* **0,088**.

Tatsächlicher Winkel (Grad)	Errechneter Winkel (Grad)	expectTwoBlobs (true/false)	Belichtungszeit
0°	0°	*true*	5
0°	1.025°	*false*	*30*
90°	90.394°	*true*	*5*
90°	88.956°	*false*	*30*
154°	153.894°	*true*	*5*
154°	152.313°	*false*	*30*

Tabelle 2: Beispiel der eingesetzten Validierungsmethodik für die Orientierung. Es wurden die Positionen aus Tabelle 1 verwendet. *Kamerahöhe***: 1,30 m**. *Binarisierungs-Schwellenwert*: **50**. *Minimale Zusammenhangkomponentengröße*: **5**. *maxLEDdistance:* **0,088**.

Das System wurde mit obigen Methoden zusätzlich im Wasser getestet, dabei wurde festgestellt, dass das Wasser das verwendete Infrarotlicht absorbiert und die Schwellenwerte dementsprechend angepasst werden müssen, die errechneten Positionen und Orientierungen trotzdem korrekt bleiben. Bei einer starken Wasserwelle werden Position und Orientierung verfälscht. Da die Fische und das Fisch-Replikat morphologisch an das Wasser angepasst sind, entstehen nur geringe Wasserwellen, die das Tracking nicht stören.

Tatsächliche Anzahl an Roboterfischen	Lokalisierte Roboterfische	*expectTwoBlobs* (true/false)	Belichtungszeit	Binarisierungs-Schwellenwert	Minimale Zusammenhangskomponenten-Größe (Pixel)	FPS
1	0	*true*	5	50	20	23
1	1	*true*	5	50	10	23
1	2	*false*	5	50	10	23
1	0	*true*	5	60	10	23
1	1	*true*	5	60	0	23
1	1	*false*	30	80	0	20
2	0	*true*	5	50	20	23
2	2	*true*	5	50	10	23
2	4	*false*	5	50	10	23
2	0	*true*	5	60	10	23
2	2	*true*	5	60	0	23
2	2	*false*	30	80	0	20
2	0	*true*	5	200	0	23
2	2	*false*	40	100	0	19
2	5	*false*	40	40	0	19

Tabelle 3: Die obige Tabelle zeigt die Ergebnisse des Trackings bei Modifikation verschiedener Schwellenwerte. *Kamerahöhe*: **1,30 m**. *maxLEDdistance*: **0,088**. Die grün eingefärbten Zeilen stellen vom Tracking korrekt erfasste Ergebnisse dar.

Tatsächliche Anzahl an Roboterfischen	Lokalisierte Roboterfische	*expectTwoBlobs* (true/false)	Belichtungszeit	Binarisierungs-Schwellenwert	Minimale Zusammenhangskomponenten-Größe (Pixel)	FPS
1	0	*true*	5	50	20	23
1	0	*true*	5	50	10	23
1	2	*false*	5	50	10	23
1	0	*true*	5	60	10	23
1	1	*true*	5	60	0	23
1	1	*false*	30	80	0	20
2	0	*true*	5	50	20	23
2	2	*true*	5	50	10	23
2	4	*false*	5	50	10	23
2	0	*true*	5	60	10	23
2	2	*true*	5	40	0	23
2	2	*false*	30	80	0	20
2	0	*true*	5	200	0	23
2	0	*false*	30	200	0	20

Tabelle 4: Die obige Tabelle zeigt die Ergebnisse des Trackings bei Modifikation verschiedener Schwellenwerte. *Kamerahöhe*: **2 m**. *maxLEDdistance*: **0,023**. Die grün eingefärbten Zeilen stellen vom Tracking korrekt erfasste Ergebnisse dar.

5 Diskussion

Das Robofish 2.0 System, das in Zusammenarbeit mit mehreren Wissenschaftlern entwickelt wird, soll bei Fertigstellung das vorhandene Robofish 1.0 System der Biologen ersetzen. Es bietet durch den Einsatz von omnidirektional fahrenden Robotern im Gegensatz zum vorherigen System den Forschern die Möglichkeit, mehrere Fisch-Replikate einzusetzen und dadurch neue Experimentierszenarien auszuprobieren.

Im Verlauf dieser Experimente wird sich herausstellen, ob sich die hier entwickelte Fisch-Nachbildung einsetzen lässt. Es wird sich zeigen, ob die verwendete Kontaktlinsenbasis lebenden Fischen auffällt und dadurch vielleicht zu groß ist. Ist dies der Fall, so kann versucht werden, diese zu verkleinern oder die Lokalisierung auf eine andere Art und Weise lösen, indem z.B. bessere Motoren (mit integriertem Feedbacksystem) beim Roboter verwendet werden.

Die Orientierungsfindung basiert auf einer Annahme, die in der Initialisierungsphase des Trackings getroffen wird. Hier muss entweder der Anwender dafür sorgen, dass die Roboter zu Beginn richtig positioniert werden, oder der Benutzer der Klasse führt eine kurze Kalibrierung nach jeder Initialisierung durch und korrigiert entsprechend die obige Annahme. Das könnte durch den Einsatz mehrerer LEDs, also mehr Information, besser automatisiert werden.

Die Validierung (siehe Abschnitt 3.5.1) hat gezeigt, dass die zweite in Ansatz 2 entwickelte Lokalisierungsmethode (siehe Abschnitt 3.4.3) ungenauer als die erste ist. Da der Einsatz dieser in bestimmten Kamerahöhen unabdingbar ist, muss diese Methode bei Fertigstellung des Reglers intensiver getestet werden. Es muss zusätzlich überlegt werden, ob das in Ansatz 2 beschriebene Problem des zu geringen Abstrahlwinkels der Infrarotlicht-LEDs anders lösbar ist, z.B. durch die Verwendung anderer Infrarotlicht-LEDs mit einem besseren Abstrahlwinkel.

In Zusammenarbeit mit den Biologen könnte in der Zukunft diskutiert werden, ob das Verfahren zur Herstellung eines Fisch-Replikates überarbeitet werden sollte (z.B. durch den Einsatz einer festen Gussform, in der die Infrarotlicht-LEDs eingesetzt werden) und dadurch besser an dieses System angepasst werden würde.

In naher Zukunft möchten Forscher die Interaktion von lebenden Fischen mit sehr kleinen Roboterfischen untersuchen. Bei der momentan verwendeten Hardware lassen sich LEDs, die in einem kleineren Abstand als 2 cm voneinander entfernt sind, bei einer Auflösung von 640x480 und einer Kamerahöhe von 1,30 m eventuell nicht korrekt lokalisieren. Sie könnten sehr stark verschmelzen, wodurch eine eindeutige Unterschei-

dung beider LEDs nicht mehr möglich wäre. Ein neues Verfahren zur Herstellung der Fisch-Replikate könnte das Problem beheben.

Würde die Kamera unterhalb des Aquariums positioniert werden, so ließen sich Infrarotlicht-LEDs statt am Fisch-Replikat direkt am Roboter befestigen, wodurch der Aufwand zur Herstellung neuer Fisch-Nachbildungen wegfällt. Das in dieser Arbeit entwickelte Tracking-System müsste dadurch nicht zusätzlich verändert werden und die Forscher könnten bereits vorhandene Fisch-Replikate für die Experimente nutzen.

6 Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde ein biomimetisches Fisch-Replikat, das an einen omnidirektional fahrenden Roboter mit starken Magneten gekoppelt und dadurch gesteuert wird, entworfen. Für die korrekte Steuerung des Roboters wurde ein Tracking-System implementiert, das mehrere Roboterfische fortlaufend lokalisieren kann. Hierfür wurden Infrarotlicht-LEDs die Fisch-Replikate eingebaut. Mit Hilfe eines entworfenen Testprogramms wurden verschiedene Experimente durchgeführt und damit die Korrektheit der durch das Tracking ermittelten Positionen und Orientierungen validiert. Zusätzlich wurde mit Hilfe eines von Hai Nguyen implementierten Reglers, der die in dieser Arbeit entwickelte Klasse verwendet hat, gezeigt, dass das implementierte Tracking für eine fortlaufende Positionskorrektur bei der Steuerung des Roboters erfolgreich verwendet werden kann.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] **Holonomic Control of a robot with an omnidirectional drive.** Raúl Rojas, Alexander Gloye Foerster. Kuenstliche Intelligenz, BoettcherIT Verlag, 2006.

[2] **Learning OpenCV: Computer Vision with the OpenCV Library.** Gary Bradsky, Adrian Kaehler. O'Reilly, 2008.

[3] **A novel method for investigating the collective behavior of fish: introducing 'Robofish'**. Jolyon J. Faria, John R. G. Dyer, Romain O. Clément, Iain D. Couzin, Natalie Holt, Ashley J. W. Ward, Dean Waters, Jens Krause. Springer-Verlag, 2010.

[4] **Social integration of robots into groups of cockroaches to control self-organized choices.** Halloy, J. et al. Science 318, 1155–1158, 2007.

[5] **A biomimetic honeybee robot for the analysis of the honeybee dance communication system**. Tim Landgraf, Michael Oertel, Daniel Rhiel and Raúl Rojas. Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3397-3103, Taipeh, Taiwan, October 18-22, 2010.

[6] **Male displays adjusted to female's response – macho courtship by the in bowerbird is tempered to avoid frightening the female.** Patricelli, G.L. et al. Nature 415, 279–280, 2002.

[7] **Quorum decision-making facilitates information transfer in fish shoals.** Ward AJW, Sumpter DJT, Couzin LD, Hart PJB, Krause J. Proc Natl Acad Sci USA, 2008.

[8] **Consensus decision making by fish.** Sumpter DJ, Krause J, James R, Couzin ID, Ward AJ. Curr Biol, 2008.

[9] **Connected Component Labeling.** R. Fisher, S. Perkins, A. Walker and E. Wolfart. http://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm, 2003.

[10] **YCC colour space and image compression.** Paul Bourke. http://paulbourke.net/texture_colour/ycc, 2000.

[11] **Computer Vision: A modern approach.** David A. Forsyth, Jean Ponce. Pearson Education International, 2003.

[12] **Pattern Recognition and Machine Learning.** Christopher M. Bishop. Springer, 2006.

[13] **Multiple View Geometry in computer vision.** Hartley Zissermann. Cambridge University Press, 2004.

[14] **Nahbereichsphotogrammetrie in der Praxis.** Thomas Luhmann. 2003.

[15] **Kamera- und Videoprojektorkalibration in medizinischen Anwendungen.** Sebastian Kreuzer. Universität Karlsruhe, 2002.

[16] **Introdction to computer vision.** Zhigang Zhu. University of Massachusetts at Amherst, 2004.

[17] **A multi-agent platform for biomimetic fish.** Tim Landgraf, Rami Akkad, Hai Nguyen, Romain O. Clément, Jens Krause, Raúl Rojas. (In Vorbereitung.)

[18] **RoboBee Picture.** Tim Landgraf. http://robobiene.mi.fu-berlin.de/img/beerobot_finalprototype.jpg.

[19] **Robotic Cockroach Picture.** http://leurre.ulb.ac.be/Announ_newsEVE.html

Anhang 1: Pseudocode für Ansatz 1

```
array my_fishes; // Die dynamische Liste mit gefundenen Roboterfischen.

array connectedComponents; // Die dynamische Liste mit gefundenen
Zusammenhangskomponenten.

while(true) do: {

getNewImageAndDoPrepocessing(); // Hole neues Bild von der Kamera und wende die
in Abschnitt 3.2 beschriebenen Verfahren an

findBlobsForRobofish(); // Suche Zusammenhangskomponenten

void findRobofishes (

     connectedComponents.clear() // Die dynamische Liste mit gefundenen
Zusammenhangskomponenten. Es wird zu Anfang gelöscht, da es als Member der Klasse
erzeugt wurde.

     findBlobsForRobofish(); // Suche Zusammenhangskomponenten (siehe Abschnitt
3.4.1) ab.

     for each (connectedComponents i) {

       bool found = false; // Wurde für die aktuelle Zusammenhangskomponente was
gefunden? Wird mit false initialisiert.

          for each (my_fishes j) {

            if(i.active==false) continue; // Es handelt sich aktuell um einen
inaktiven Roboterfisch. Dieser muss nicht behandelt werden und wird übersprungen.

            if(distance j to i < maxFishMovement) i belongs to j ->
(update positions, update ttl, update history, found = true, break) //
Die Distanz von der aktuellen Zusammenhangskomponente zur letzten Position des aktuellen
Roboterfisches ist kleiner als maxFishMovement, daher muss element zu fish gehören und
eine Zuordnung wurde gefunden.

          }

       if(found==false) i is a new fish -> add to my_fishes // Es wurde
keine Zuordnung für element gefunden, daher handelt es sich um einen neuen Roboterfisch,
der in die Roboterfisch-Liste hinzugefügt wird.

);

my_fishes.decrease_TTL // Verringere für jeden Roboterfisch die TTL um 1. Ist sie
0, so setze active=false.

}
```

Anhang 2: Pseudocode für *findLedPairs()*

```
array currentPairs; // Zur Erinnerung: Die dynamische Liste, welches die LED Paare enhält.

array connectedComponents; // Zur Erinnerung: Die dynamische Liste mit gefundenen Zusammenhangskomponenten.

void findLEDPairs() {

  currentPairs.clear(); // Die dynamische Liste, welches die LED Paare enhält. Es wird zu Anfang gelöscht, da es als Member der Klasse erzeugt wurde.

  for each (connectedComponents i)

  {

    if (i.used) continue; // Die aktuelle Zusammenhangskomponente wurde bereits einer anderen LED zugeordnet.

    for each (connectedComponents j=i+1) { // Durchsuche alle nach i folgenden Zusammenhangskomponenten in connectedComponents, ob sie zur Zusammenhangskomponente i passen.

      if (j.used) continue; // Die aktuelle Zusammenhangskomponente wurde bereits einer anderen LED zugeordnet.

      if ( distance(i, j) < maxLEDdistance) // Die beiden Zusammenhangskomponenten i und j haben eine passende Distanz und gehören zu einem Fisch-Replikat.

      {

        i.used = true; // Markiere Zusammenhangskomponente i als verwendet.

        j.used = true; // Markiere Zusammenhangskomponente j als verwendet.

        position = (i+j)/2; // Als Position eines LED-Paares wird der Punkt zwischen zwischen i und j genommen.

  currentPairs.add(i,j,position) // Füge i und j als ein Paar zur LED_PAIR Liste hinzu.

      }

    }

  }

}
```

Anhang 3: Pseudocode für *findFirstFishes()*

```
array currentPairs; //Zur Erinnerung: Die dynamische Liste, welches die LED Paare enhält.

array my_fishes; // Zur Erinnerung: Die dynamische Liste mit gefundenen Fischen.

void findFirstFishes()

{

   getNewImageAndDoPrepocessing(); // Hole neues Bild von der Kamera und wende die in Abschnitt 3.2 beschriebenen Verfahren an.

   findBlobsForRobofish(); // Finde Zusammenhangskomponenten.

   my_fishes.clear(); // Lösche die Liste, die alle gefundenen Roboterfische .enhält.


   for each (currentPairs i)

   {

      my_fishes.add(i); // Erzeuge neuen Roboterfisch für jedes Element aus currentPairs.

   }

}
```

Anhang 4: Pseudocode für Ansatz 2

```
array my_fishes; // Zur Erinnerung: Die Liste mit gefundenen Fischen.

array connectedComponents; // Zur Erinnerung: Die dynamische Liste mit gefundenen Zusammenhangskomponenten.

array currentPairs; //Zur Erinnerung: Die dynamische Liste, welches die LED Paare enhält.

while(true) do: {

getNewImageAndDoPrepocessing(); // Hole neues Bild von der Kamera und wende die in Abschnitt 3.2 beschriebenen Verfahren an

void findRobofishes (

    connectedComponents.clear() // Das dynamische array mit gefundenen Zusammenhangskomponenten. Es wird zu Anfang gelöscht, da es als Member der Klasse erzeugt wurde.

    findBlobsForRobofish(); // Suche Zusammenhangskomponenten.

    findLEDPairs(); // Suche zusammenhängende LEDs.

    for each (my_fishes i) and for each (currentPairs j) {

      find led_pair in currentPairs with smallest distance and set a found flag for i related to j // Suche zu allen i zugehörige LED-Paare j, dabei wird immer der kürzeste Abstand verwendet. Jedes i wird mit jedem j verglichen. Falls ein späteres i+x einen besseren (=kleineren) Abstand zu einem bereits zugeordneten j hat, so wird die Zuordnung von i geändert und der vorherige Fisch wird hinter i+x in my_fishes verschoben, um erneut von der for Schleife erfasst werden zu können.

    }

    for (each my_fishes i) {

      if (i has related LED_PAIR) do { // Wurde für den aktuellen Roboterfisch i ein zugehöriges LED-Paar gefunden?

        angle one = angle of LED_PAIR; // Berechne Winkel für die beiden Punnkte in LED_PAIR. LED_PAIR besteht aus zwei Punkten a und b, es wird angenommen, dass a vorne und b hinten ist.

        angle two = old angle of i; // Winkel zwei wird die alte, noch nicht aktualisierte Winkel von Fisch i.

        angle diff = |two – one|; // Berechne den Betrag Differenz.

        if (diff > PI) diff = 2*PI – diff; // Falls two < one ist, würde der Betrag der Differenz beider Winkel keinen gültigen Winkel ergeben und muss entsprechend korrigiert werden.

        if (diff > PI/2) i.related_LED_PAIR.swapPoints(); // Schaue, ob der neue Winkel "Sinn" macht (siehe obige Erklärung). Falls nicht, müssen in LED_PAIR die Punkte vertauscht werden (a wird b und b wird a).
```

```
          update position, angle, ttl, history for i; // Aktualisiere
Position, Winkel, History, usw. für aktuellen Roboterfisch, wobei a aus LED_PAIR =
vorderer Punkt, b aus LED_PAIR = hinterer Punkt und position aus LED_PAIR = position
wird.

        }

      }

);

my_fishes.decrease_TTL // Verringere für jeden Fisch die TTL um 1. Ist sie 0, so
setze active=false.

}
```

Anhang 5: Beispielimplementierung der Tracking-Klasse

```
rectification(); // Rektifiziere zuerst den zu trackenden Bereich (gemäß Abschnitt 3.3).

setThreshold(50); // Setze den Schwellenwert für Binarisierung auf 50.

setAutoMaxLEDCalculation(true); // Aktiviere die automatische Berechnung von maxLEDdistance (gemäß Abschnitt 3.4.3 ).

setMinBlobSize(5); // Setze die minimale Zusammenhangskomponentengröße auf 5 Pixel (gemäß Abschnitt 3.4.1).

setExpectTwoBlobs(true); // Für zwei in einem Fisch-Replikat verbaute Infrarotlicht-LEDs werden zwei Zusammenhangskomponenten erwartet (gemäß Abschnitt 3.4.3).

findFirstFishes(); // Initialisiere das Tracking (gemäß Abschnitt 3.4.3)

if (my_fishes.size() == 2) { // Zwei Roboterfische sollen getrackt werden. Daher wird überprüft, ob die oben gesetzten Schwellenwerte korrekt sind und damit zwei Roboterfische gefunden wurden.

  run(); // Startet das Threading. Der Thread wendet die in Abschnitt 2.1 beschriebenen Schritte und Algorithmen in jedem Durchlauf an und lokalisiert ununterbrochen die Roboterfische.

}

while (all fishes are active) // Der gestartete Thread läuft so lange, bis das Tracking einen Roboterfisch verloren hat.

{

  printMyFishes(my_image); // Zur Visualisierung werden alle Fische in ein Bild einzeichnen und dem Benutzer in einem Fenster angezeigt.

  do_something_with(my_fishes[0].position,my_fishes[0].orientation); // Mit dem ersten lokalisierten Roboterfisch aus der Liste wird gearbeitet.

}

quit(); // Das Tracking wird nicht mehr benötig und der oben gestartete Thread wird beendet.
```