Freie Universität Berlin
Fachbereich Mathematik und Informatik

Institut für Bioinformatik

Bachelorarbeit

Implementierung eines Partikelfilters für das Tracking von Fischen

Angelika Szengel

Erstgutachter: Prof. Dr. Raúl Rojas

Zweitgutachter: Dr. Hamid Mobalegh

Betreuer: Dr. Tim Landgraf

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und ohne unerlaubte Hilfsmittel angefertigt habe.
Ich versichere, dass ich keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche in der Arbeit gekennzeichnet habe.

1.5.2014 Angelika Szengel

Inhaltsverzeichnis

Motivation....4
State of the Art....6
Implementierung....10
Initialisierung....11
Bewertung der Partikel....12
Das Importance-Resampling....12
Positionsermittlung mit Hilfe von Clustern....15
Evaluierung....17
Diskussion....24
Ausblick....26

Motivation

Viele Tiere bilden Schwärme, um ihre Überlebenschancen zu steigern. Ein Schwarm besteht aus n-vielen Individuen. Diese agieren und reagieren untereinander mit Hilfe eines kollektiven Verhaltens. Die Summe aller Reaktionen auf Innenreize von den Individuen und Außenreize aus der Umwelt bestimmt das Auftreten des ganzen Schwarms.
Um das Schwarmverhalten zu erforschen, gibt es mehrere Ansätze.
Die klassische Verhaltensforschung basiert auf Beobachtungen mit kontrollierten Bedingungen und wenig variablen Parametern. Durch diese können Hypothesen aufgestellt werden, die wiederum durch das systematische Verändern von den gegebenen Parametern getestet werden. Der Einfluss auf die Tiere und das anpassen des System sind jedoch nur beschränkt möglich.[1] In der theoretischen Biologie werden künstliche Schwärme anhand von bekannten Informationen und Theorien modelliert. Das Modell eines Individualverhaltens, das beispielsweise ein bestimmtes Geschehen im Schwarm erklären soll, wird im Computer simuliert. Dessen Ergebnisse werden mit den in der Natur beobachteten Verhaltensweisen verglichen. Da diese Versuche unabhängig vom natürlichen Schwarm stattfinden, sind die Möglichkeiten zum Anpassen des Models eingeschränkt, da die direkte Rückkopplung zu den Individuen fehlt.[2]
Ein moderner Ansatz ist das einschleusen eines oder mehrerer biomimetischer Roboter in einen echten Schwarm. Dem Roboter wird ein Verhaltensmodell zugewiesen, welches er unter natürlichen Bedingungen ausführt. Es existiert ein direkter Vergleicht zwischen Modell und Realität. Schwierigkeit hier ist die Akzeptanz der Tierattrappe als vollwertiges Mitglied des Schwarms. Erst dann kann die eigentliche Forschungsarbeit beginnen.
Robofish verwendet diesen dritten Ansatz, um eine Fischattrappe in einen Fischschwarm zu integrieren. Die Fische akzeptieren den Roboter und das System misst das individuelle Verhalten der Tiere. Das Verhaltensmodell des Roboterfischs kann nun spezifisch an das beobachtete Verhalten der einzelnen Individuen angepasst werden.
Ein wichtiger Teil von Robofish ist das Fisch-Tracking. Es wird in zwei Kontexten eingesetzt: Bei den Versuchen im laufenden Betrieb des Roboters soll es alle Fische und den Roboter erkennen, verfolgen und jedem eine eindeutigen ID zuweisen. Während eines laufenden Experiments werden diese Daten dann an den Roboter gesandt. Er kann damit sein eigenes Verhalten dem der Fische anpassen, beispielsweise um sie zu verfolgen. Es handelt sich dabei um ein sogenanntes geschlossenes Feedback-System.
Zweitens soll das Trackingsystem auch auf Videodaten angewandt werden, in denen nicht unbedingt auch Roboter im Einsatz sind. Biologen, die mit dem Biorobotics Lab kooperieren, verwenden dieses Programm zur klassischen Verhaltensanalyse von Fischen. Beobachtungen über jedes einzelne Individuum und dessen Einfluss auf dem Schwarm werden möglich.
Für eine eindeutige und fehlerfreie Erkennung muss das Tracking folgende Anforderungen erfüllen:

1. Erkennung aller Fische

2. stabile Verfolgung aller sich bewegender Fische
3. eindeutige Zuordnung der ID, kein ID-Wechsel zwischen Fischen
4. keine bis einzelne Detektionsverluste eines Fisches
5. keine falschen Detektionen
6. trackt nicht außerhalb eines gewünschten Suchbereichs
7. robust gegen geringfügige Veränderungen wie leichte Lichtfluktuation, leichte Wasserbewegungen

Derzeit wird hierzu eine Methode verwendet, die in den Punkten 1, 2, 3 und 4 keine oder nur eingeschränkt zufriedenstellende Ergebnisse liefert. Näheres dazu wird im Abschnitt „State of the Art“ erläutert.
Sequenzielle Monte-Carlo-Methoden [3] modellieren für einen gegebenen Zustandsraum eine Wahrscheinlichkeitsverteilung. Dieser probabilistische Ansatz verspricht eine flexiblere Problembewältigung, da die Verteilung durch jedes einzelne Bild beeinflusst wird, und somit ein besseres Ergebnis als die verwendete deterministische Methode, welche nur die alten Positionen aus dem vorherigen Bild mit den neuen Positionen aus dem aktuellen Bild vergleicht und zuordnet.
Diese Bachelorarbeit hat sich zum Ziel gesetzt, das Fisch-Tracking zu verbessern. Es wurde ein Partikelfilter implementiert werden und eine geeignete Methode, eine Stichprobe aus dem Zustandsraum zu bewerten. Dabei wurde ein generisches Design die Anwendbarkeit des Filters in anderen verwandten Projekten ermöglichen.

State of the Art

Das Robofish-System besteht aus mehrere Hardware- und Software-Komponenten.

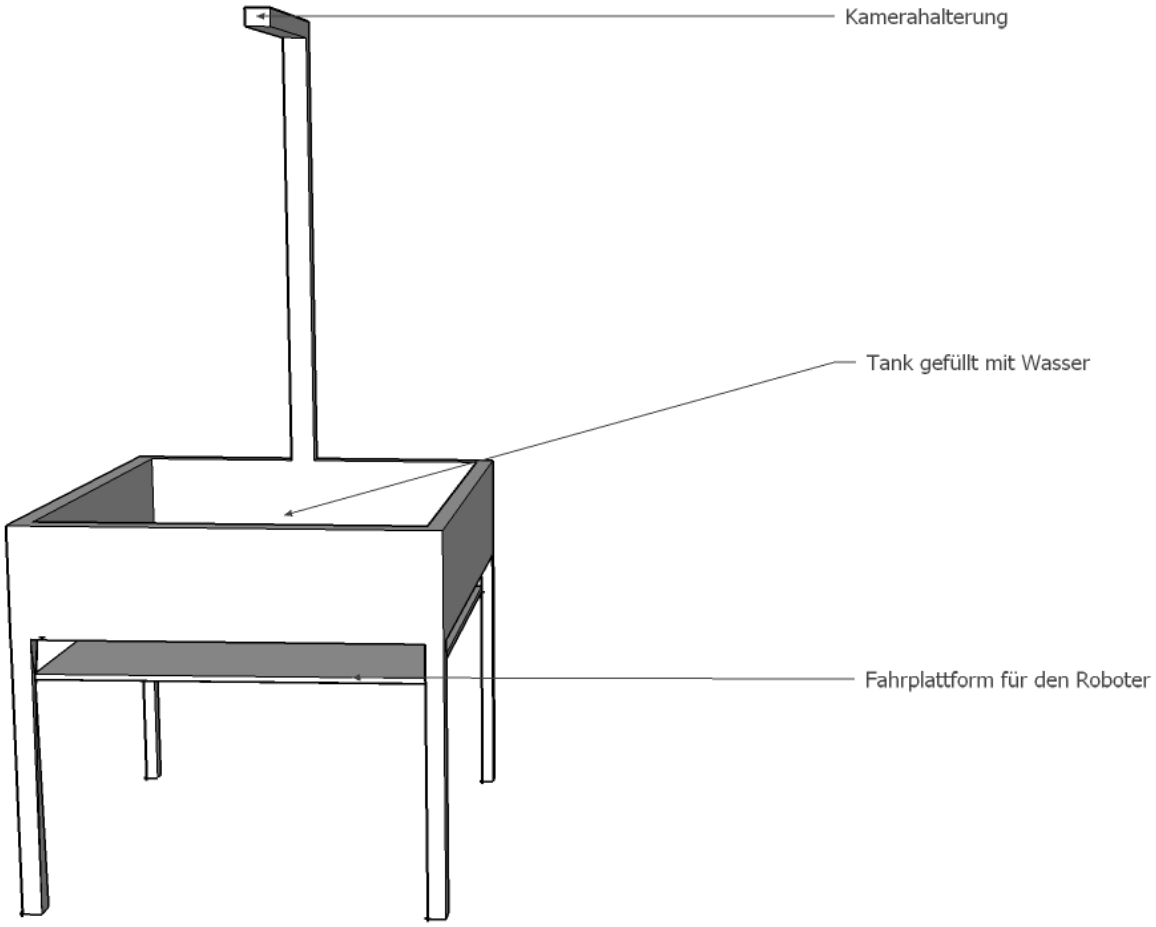

Abbildung 1: Abbildung 1: Der Versuchsaufbau von Robofish.Für die Fische gibt es einen Wassertank mit einer Grundfläche von 1 x 1 Metern. Direkt darunter befindet sich eine Ebene, auf der der Roboter fahren kann. Durch Magnete steht der Roboter mit einer Fischattrappe in Verbindung und kann diese durch den Tank fahren.Von unten filmt eine Kamera den Roboter zur Positionsermittlung. Von oben hält eine zweite Kamera das Geschehen im Wassertank fest. Die Software wird auf zwei unterschiedlichen Computern ausgeführt. Der eine Computer verwendet die RoboGUI. Mit dieser kommuniziert der Roboter und lässt sich damit steuern und verwalten. Auf dem anderen Computer läuft der FishTracker, der die Fische verfolgt und mit IDs versieht. Die RoboGUI bekommt vom FishTracker die Informationen über die Fische bereit gestellt.

Beim Tracking wird im ersten Schritt eine Backgroundsubstraction ausgeführt: Die Differenz zwischen einen Referenzbild, auch Hintergrundbild genannt, mit dem gegenwärtig auszuwertenden Bild lässt unterschiedliche Bildbereiche erkennen, welche auf Bewegung hinweisen. Das Bild wird binarisiert, sodass alle veränderten Bildbereiche weiß eingefärbt werden beziehungsweise den Wert 1 besitzen. Alle

unveränderten Bereiche bekommen den Wert 0 und sind schwarz.
Im besten Fall würden nur die Fische als weiße Bereiche zurück bleiben. Fehler können jedoch entstehen durch:

1. schlechte Lichtverhältnisse
2. Störkörper im Versuchsaufbau
3. Wasserbewegungen
4. Schattenwurf

Fehlerquellen 1 und 2 müssen während des Versuchsablaufs beseitigt werden. Wenn Fehler 1 nicht allzu gravierend ist, kann er mit Fehler 2 durch folgende morphologische Operationen ausgebessert werden: Die Erosion setzte einen Pixel von eins auf null, wenn mindestens ein Nachbarpixel null ist [4] und soll die Störungen herausfiltern. Es wird angenommen, dass weiße Bildbereiche, die Fische repräsentieren, groß genug sind, um durch diesen Prozess nicht zu verschwinden. Mitunter ist der Übergang von Fischschwanz zum Körper sehr eng und kann durch Erosion zwei unzusammenhängende Flächen entstehen lassen. Dagegen soll die Dilatation helfen, welche Pixel von null auf eins setzten, wenn mindestens ein Nachbarpixel eins ist.[4] Die weißen Bildbereiche werden wieder größer. Versehentlich getrennte Flächen werden wieder vereint. Problematisch sind sehr kleine Fische. Diese können durch die Erosion komplett verloren gehen und zumindest auf diesen Bild nicht mehr auffindbar sein.
Da der Kameraausschnitt größer ist als der Wassertank, muss auf dem Bild eine Sucheinschränkung erfolgen. Eine Maske ist ein Bild, auf dem der Suchbereich als ein weißes Polygon auf schwarzem Hintergrund markiert ist. Dieses Bild wird mit dem Kamerabild verglichen. Alle schwarzen Flächen auf der Maske werden im Kamerabild beim Tracking nicht mehr beachtet. Störungen in der Umgebung des Tanks sind somit eliminiert. Die Methode verhindert jedoch keine Störfaktoren, die direkt im Bereich des Tanks liegen.
Die übrig gebliebenen, weißen Bildbereiche werden Blobs genannt. Um die Eigenschaften der Blobs in Form von Position, Winkel und Größe zu extrahieren, wird ein Blobdetektor eines Drittanbieters verwendet.
Die gegenwärtige Standardmethode für die Zuordnung der Fisch-IDs zu den Blobs ist „Nearest Neighbours“. Dabei werden Fischpositionen aus dem vorherigen Bild mit den Fischpositionen aus dem aktuellen Bild verglichen. Die Positionen mit dem geringsten euklidischen Abstand aus allen möglichen Fällen werden einander zugeordnet und die alten ID-Nummern werden dementsprechend verteilt.

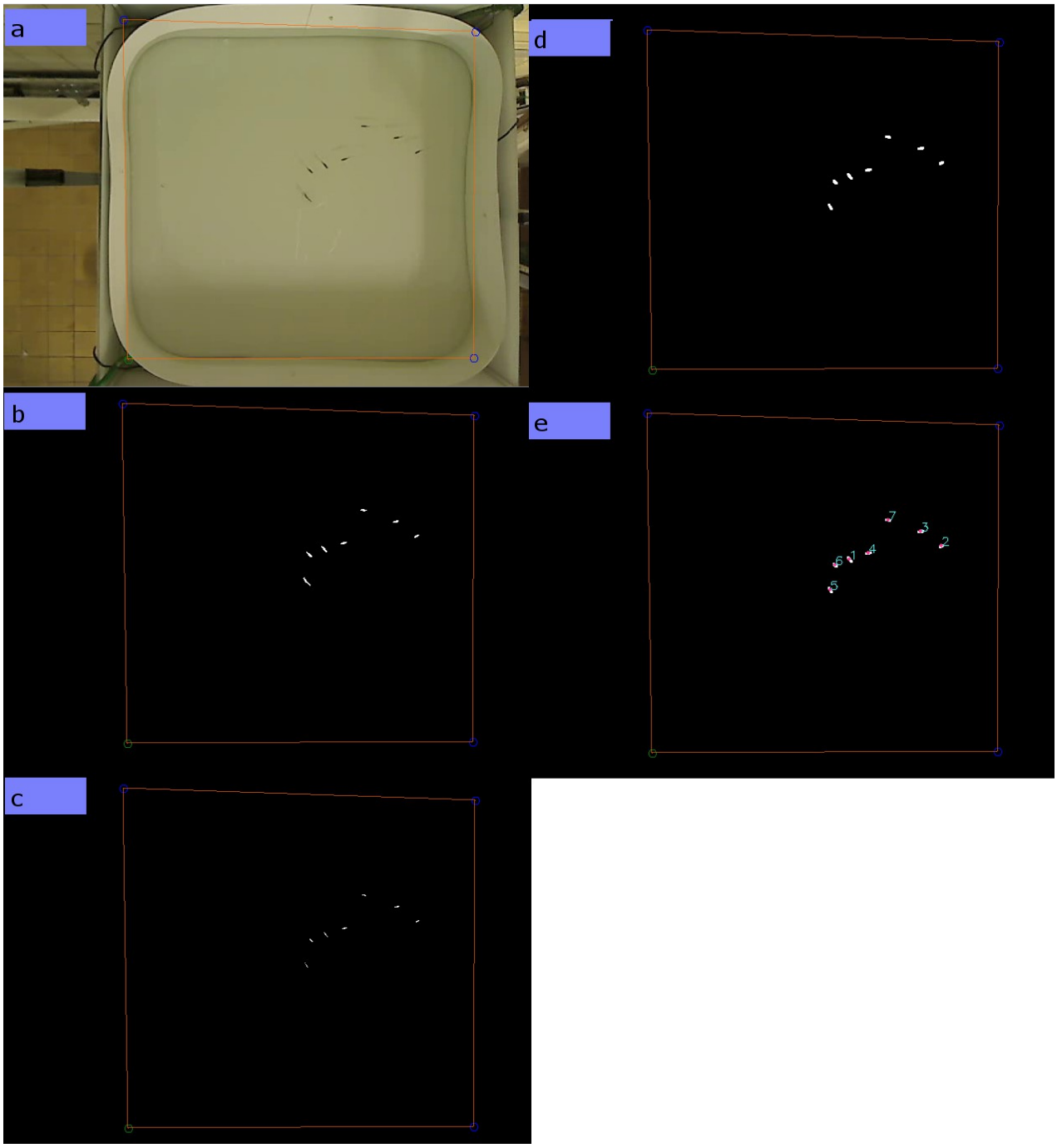

Abbildung 2: Bildbearbeitungsprozess für das Tracking

a) Das Originalbild der Kamera. Fische, Tank, und Umgebung sind darauf zu erkennen. Eine zusätzliche Wand schirmt die Fische von den Tankecken ab. Das orangene Viereck kennzeichnet den Trackingbereich. b) Das binarisierte Bild mit Hilfe der Backgroundsubstraction. Die sich bewegenden, dunklen Fische auf hellem Hintergrund übersteigen den Schwellwert beim Binarisieren und werden auf 1 gesetzt. Dadurch entshen die hellen Bildbereiche, hier auch Blobs genannt. c) Erosion wurde durchgeführt. Die Blobs verlieren Ranfläche und werden kleiner. Zu kleine, weiße Flächen vom Rauschen gehen verloren. d) Dilatation wurde durchgeführt. Die Blobs erhalten mehr Randfläche und vergrößern sich. e) Beispielhaft zugewiesene ID's mit Hilfe der Nearest Neighbours Methode.

Dabei kann es jedoch passieren, dass die IDs den falschen Fischen zugeordnet werden und dadurch springen. Dies passiert vor allem, wenn die Fische zu eng zusammen Schwimmen. Der Abstand zum Nachbar ist dann so gering, dass keine Unterscheidung mehr getroffen werden kann. Blobs können im Binärbild verschmelzen und als eine zusammenhängende Einheit erkannt werden. Die gegenwärtige Methode bekommt als Information, wie viele Fische fest zu verfolgen sind. Es wurde jedoch nie implementiert, wie verschmelzende und sich wieder trennende Blobs ausgewertet werden sollen. Verlorene Positionen bleiben lediglich auf ihrer letzten, ermittelten Stelle zurück, bis die verlorenen Fische über den Blobdetektor wieder erkannt werden.
Es gibt auch den Spezialfall, dass sich Fisch A im nächsten Bild auf der Position von Fisch B befindet. Fisch A hätte dann zu Fisch B den Abstand Null und würde dessen ID bekommen. Das passiert jedoch eher, wenn die Bildrate zu niedrig ist und sollte beim normalen Betrieb nicht auftreten.

Es gibt zwei weitere Projekte, welche ebenfalls das Schwarmverhalten von Fischen untersuchen. Beide verwenden einen sehr ähnlicher Versuchsaufbau mit einem Wassertanks und einem Roboter, der eine Fischattrappe steuert. Sie benutzen jedoch nur eine Kamera für das Filmen von Roboter und das Geschehen im Tank und verwenden nur einen Computer für ihre Software.
Bonnet et al. verwenden Zebrafische und versuchen ebenfalls eine Attrappe in den Fischschwarm zu integrieren. Swain et al. hingegen verwenden für den Roboter einen Räuberfisch, der auf die „Golden Shiners“-Fische zusteuert und untersuchen das daraus resultierende Räuber-Beute-Verhalten.
Für das Tracking verwenden beide ebenfalls das Backgroundsubstraction, um Blobs heraus zu filtern. Zur weiteren Auswertung benutzt Swain jedoch einen “unscented Kalmanfilter” (zitat). Bonnet stützt sich auf die Tsai Kallibrationstechnik (zitat).
Beide Projekte waren 2012 noch in der Testphase. Swain hatte jedoch schon einige Möglichkeiten mit dessen System demonstriert und mit Aufnahmen dokumentiert.[5][6]
Da deren Quelltext nicht öffentlich zugänglich ist, gab es keine Möglichkeiten, deren Arbeiten für das Robofish-Projekt zu verwenden. Dadurch musste die Software komplett neu geschrieben werden.

Implementierung

Das Ziel war es, einen wieder verwendbaren Partikelfilter zu schreiben. Die Struktur durfte daher nicht zu sehr mit dem gesamten Projekt verstrickt sein. Die Bewertung der Partikel soll austauschbar sein. Ergebnisse über getrackte Objekte werden gesondert an externe Systeme zurückgegeben und können dort in gewünschte Formate umgewandelt werden. Dadurch kann der Partikelfilter auch in anderen Szenarien eingesetzt werden. Gleichzeitig verschafft es die Möglichkeit, auch innerhalb des Robofish-Projekts den Filter schnell an neue Situationen oder Ideen anzupassen beziehungsweise zu verbessern.
Im Robofish-Szenario müssen die Fische ohne Ausfälle getrackt werden. Sollte zwischenzeitlich trotzdem ein Fisch kurz verloren gehen, hat der Partikelfilter diesen so schnell wie möglich wieder zu finden. IDs müssen immer korrekt zugeordnet werden. Die Laufzeit sollte schnell genug sein, sodass im laufenden Betrieb die Bildrate nicht unter zwanzig Bilder pro Sekunde (FPS) fällt.
Der Filter verwendet Partikel, um die gesuchten Objekte zu lokalisieren. Ein Partikel verkörpert einen Punkt im Zustandsraum, der beim Fisch-Tracking aus kartesischen Koordinaten auf einer planaren Fläche, Orientierungswinkel und gegebenenfalls der Größe der Individuen besteht. In diesem Raum werden Partikel gestreut. Jedem Partikel wird eine Bewertung gegeben, die proportional zur Wahrscheinlichkeit der Messung im Kamerabild ist. Das geschieht unter der Annahme, dass das gesuchte Objekt tatsächlich so konfiguriert ist, wie im Partikel kodiert. In einem “Importance-Resampling” genannten Schritt wird nur der Teil der Partikel weiter verwendet, der hinreichend gut bewertet wurde. Partikel mit guter Bewertung werden häufig ausgesucht und vervielfältigt, diejenigen mit schlechter Bewertung “sterben aus”.
Der Filter ist objektorientiert und wurde in der Programmiersprache C++ geschrieben.

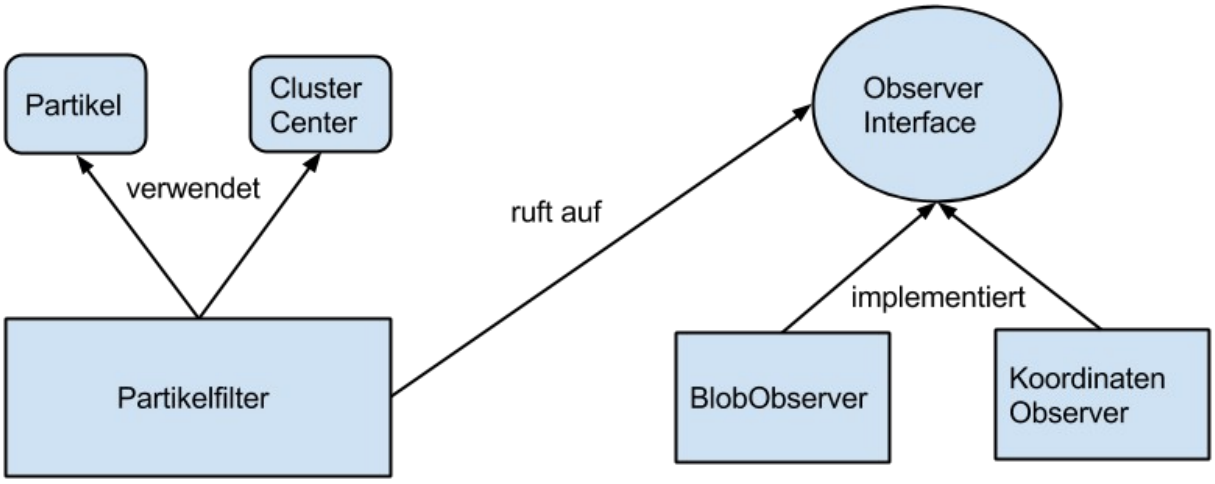

Abbildung 3: Abstrakte Übersicht zu den einzelnen Klassen des Partikelfilters

Die Klasse *Partikel* speichert wichtige Informationen aus dem Zustandsraum. Gegenwärtig sind das die x- und y-Koordinaten, der Ausrichtungswinkel in Grad und die Bewertung, hier Belief genannt. Es gibt noch die Eigenschaften der Höhe und Breite, sie werden aber gegenwärtig noch nicht verwendet. Einige Hilfsfunktionen erleichtern das Rechnen mit den Partikeln. Diese Klasse kann bei weiteren oder abweichenden Anforderungen erweitert werden.
Der *Observer* prüft jedes Partikel auf jeden gefundenen Blob und gibt die beste Bewertung an den Partikelfilter zurück. Zur Ermittlung der Blobs wird der schon vorhandene Blobdetektor verwendet. Der Partikelfilter fragt ebenfalls den Observer an, in welchen Bereich des Bildes er die Partikel streuen darf. Durch ein Interface ist der Observer durch andere Methoden austauschbar.
Der *Partikelfilter* initialisiert die Partikel, holt sich die aktuellen Bewertungen vom Observer, resampelt darauf basierend die Partikel und clustert die entstehenden Partikelwolken, um die Positionen zu ermitteln und zu speichern. Diese können bei Bedarf abgefragt werden.
Clustercenter ist eine Klasse, die berechnete Cluster speichert. Das Zentrum, repräsentiert als Partikel, entspricht einer gefundenen Position. Während des Clusterns werden zugeordnete Partikel mit dem Zentrum verrechnet. Eine ID wird vergeben, die gegenwärtig ebenfalls der Bezeichner für die Position ist und die ID aller zugehörigen Partikel für den nächsten Durchlauf festlegt.
Für Robofish kontrolliert ein Partikelfiltertracker, wann der Partikelfilter welchen Observer verwenden soll, um die Fische zu tracken. Dieser wandelt auch die Informationen der Clusterzentren zur weiteren Verarbeitung in Fischpositionen um.

Die Arbeit des Partikelfilters besteht auf mehreren Abschnitten:

- Initialisierung
- Bewertung der Partikel durch den Observer
- Importance-Resampling
- Positionsermittlung mit Hilfe von Clustern

Initialisierung

Der Filter wird im Konstruktor mit Standardwerten vom System initialisiert.
Der Partikelfilter vergibt alle vorgegebenen Partikel zufällige Werten und streut sie so im Zustandsraum. Er fragt den Observer an, in welchen Intervallen die Werte sich befinden dürfen. Dadurch werden die Partikelkoordinaten über den ganzen Wassertank gestreut und Winkel erhalten ihr Gradmaß zwischen 0 und 360°. Jedes Partikel bekommt ebenfalls eine eigene ID. Dieser Abschnitt ist in der derzeitigen Implementierung nicht generisch und kann keine Partikel mit abweichenden Attributen behandeln.
Um dem Partikelfilter die Arbeit zu erleichtern, gibt es die Funktion, einige Partikel direkt um die vorhandenen Blobs zu streuen. Dadurch kann schneller das Maximum gefunden werden und die Problematik wird verkleinert, dass einige aufzufindende Blobs zu ungünstig liegen und dementsprechend

nicht entdeckt werden, da keine Partikel in direkter Nähe sind.

Bewertung der Partikel

Innerhalb von update() wird jedes Partikel für seine Bewertung, auch Belief genannt, an den Observer geschickt. Wenn jeder Partikel einen Belief besitzt, werden diese normalisiert. Alle Werte befinden sich dann zwischen Null und Eins.

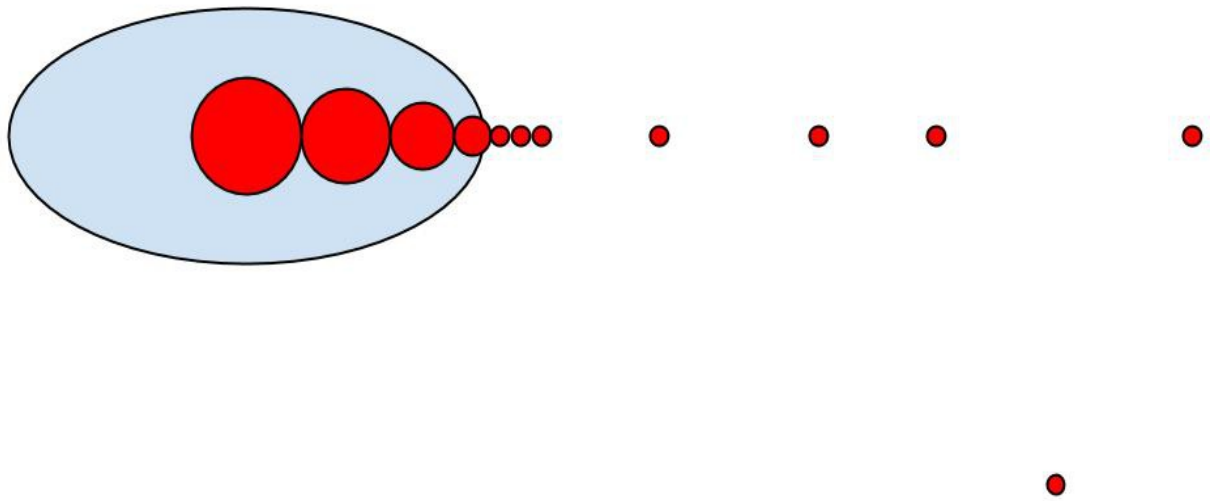

Abbildung 4: Abhängigkeit des Beliefs symbolisiert durch die Größe des roten Partikels. Je näher der Partikel sich am Mittelpunkt der Ellipse befindet, desto größer wird sein Belief. Je nachdem wie hoch die Varianz im Observer eingestellt wurde, fällt der Belief unterschiedlich schnell ab. In diesem Bild fällt der Belief eines Partikel im Idealfall am Rand der Ellipse auf eine Restwahrscheinlichkeit herunter. Da die Blobs nur Messungen mit zu erwartenden Messfehlern sind, soll die Restwahrscheinlichkeit die Möglichkeit offen halten, dass sich ebenfalls woanders Fische aufhalten. Es gab die Idee, dass ihr die Cluster von Blobs möglicherweise bei fehlenden Detektionen länger überleben könnten, bis das verlorene Objekt wieder gefunden und der Cluster neu zugeordnet wurde.Gegenwärtig fällt der Belief kreisförmig um den Mittelpunkt ab und bewirkt somit eine unsaubere Abgrenzung des ellipsoiden Fischen gegenüber der Umwelt.

Das Importance-Resampling

In resample() wird ein Histogramm für die Partikel-IDs erzeugt. Es wird eine kumulative Summe aus den Beliefs aller Partikel gebildet. Diese symbolisiert die Grenzen zwischen den einzelnen Partikelbeliefs.

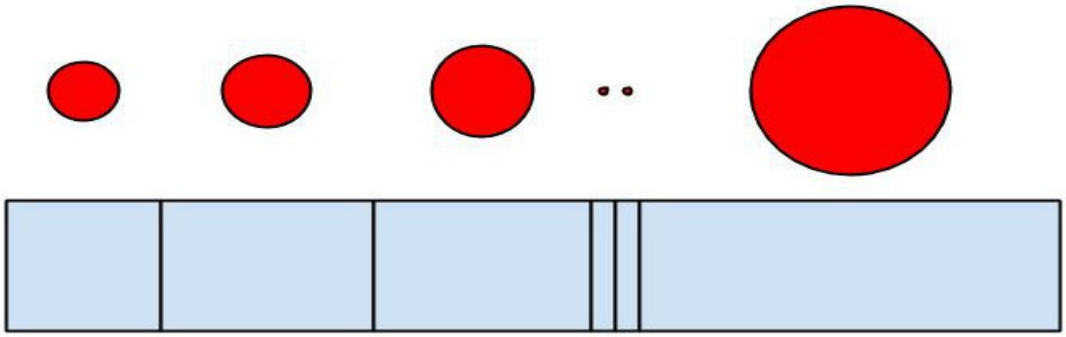

Abbildung 5: Kumulative Summe

Um ein zufälliges Ziehen mit zurücklegen der Partikel in Abhängigkeit ihrer Beliefs zu simulieren wird eine kumulative Summe gebildet. Die Werte dieser Summe bilden die Grenzen zwischen den unterschiedlich großen Partikeln. Zwischen diesen Grenzen liegt ein Partikel. Je größer dieser Bereich, desto größer ist der Belief des betroffenen Partikels. Eine zufällige Zahl zwischen 0 und dem Maximum der Summe wird durch einen Größer-gleich-Vergleich häufiger diese großen Bereiche und damit Partikel mit höhere Beliefs treffen. Der Index des getroffenen Eintrags entspricht dem getroffenen Partikel im ursprünglichen Vektor.

Dadurch kann eine Zahl zwischen Null und dem Maximum der Kumulativen Summe gewürfelt und mit einer Binärsuche im Array ermittelt werden. Der gefundene Index entspricht dem zufällig gezogenen Partikel. Die ID jedes getroffenen Partikels wird in dem Histogramm an dem gleichzahligen Index aufaddiert.

Das Resampling führt schnell dazu, dass sich alle Partikel an lediglich einer Position sammeln. Es entsteht das Problem, dass sich nur wenige Partikel an den anderen gesuchten Positionen vermehren. Sie sind in der Minderheit und haben selbst mit guten Belief schlechtere Trefferchancen als die Mehrheit der ebenfalls gut bewerteten Partikeln. Somit können bei mehreren Objekten im Suchfeld einige potentiell nicht gefunden werden.

Als Lösung bekommt jeder Partikel bei der Initialisierung eine einzigartige ID zugewiesen. Da die IDs beim Resampling unverändert kopiert werden, vermehren sich die IDs von Partikeln mit gutem Belief. Diese lokalen Anhäufungen von Partikeln mit derselben ID wird als eine ID-Familie bezeichnet. Man beschränkt diese so entstandenen ID-Familien auf eine maximale Anzahl im Histogramm. Wird dieses Maximum erreicht, darf der Partikel nicht wieder verwendet werden und der Belief bei dem betroffenen Partikel wird verkleinert, um die Trefferchance zu verringern. Dabei muss die kumulative Summe immer wieder neu berechnet werden. Es wird nach einen anderen geeigneten Kandidaten gesucht. Somit bekommen auch andere Partikel mit gleich guten und leicht schlechteren Beliefs die Chance der Wiederverwendung. Es können sich Partikel an anderen Stellen häufen. Falls dieser Vorgang jedoch zu häufig passiert, gibt es anscheinend keine guten Partikel mehr. Der Resamplingprozess wird abgebrochen und die restlichen Partikel werden neu gestreut. Ohne diese Abbruchbedingung würde das weitere Abstrafen guter Partikel die Unterschiede zu den eigentlich schlecht bewerteten Partikel aufheben. Dadurch könnten sich Partikel an

falsche Positionen häufen.

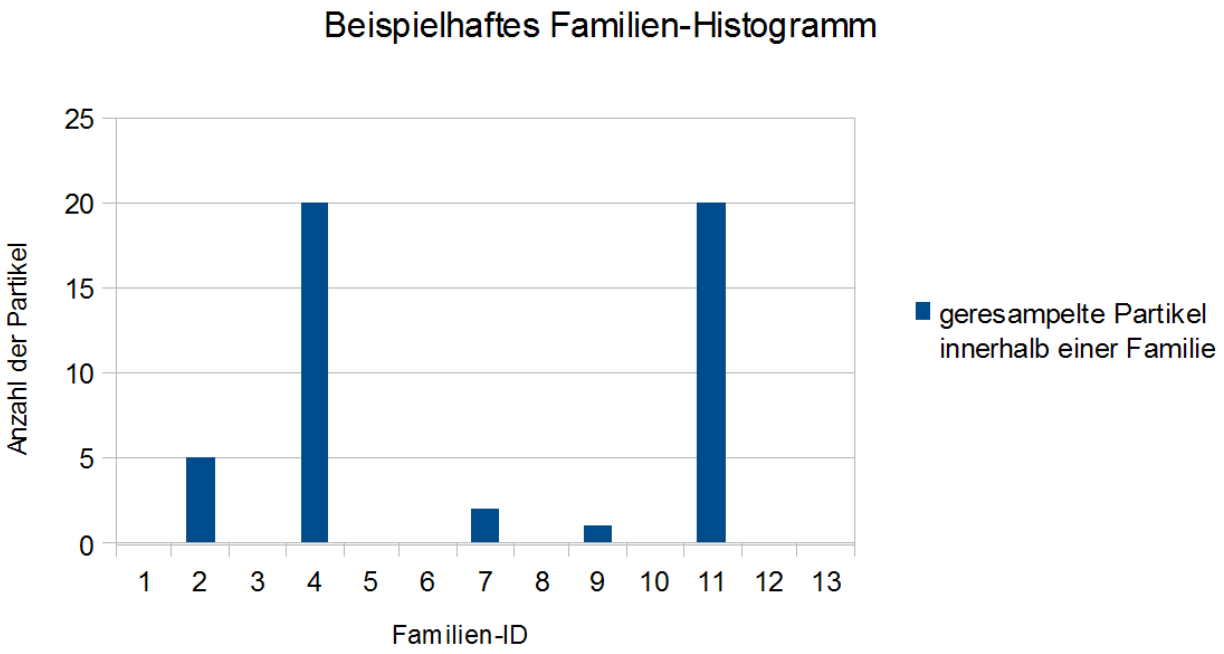

Abbildung 6: Abbildung 6: Beispielhaftes Histogramm von ID-Familien. Auf der x-Achse befindet sich die Familien-ID. Auf der y-Achse die Anzahl der dazugehörigen Partikel. Wenn ein Partikel wiederverwendet wird, verändert sich seine vorher bestimmte ID nicht. Dadurch sammeln sich mehrere Partikel mit leicht unterschiedlichen Koordinaten und aber gleicher ID. Alle diese Partikel gehören zu einer Familie. Der Bezeichner dieser Familie ist die Nummer der ID. Dadurch, dass es nur wenige Partikel in der Nähe von Ellipsen gibt und dementsprechend gute Beliefs besitzen, gibt es nur einige große Familien und einige wenige kleine. Da die meisten Partikel nicht wiederverwendet werden, hat ein Großteil dieser Familien eine Gesamtgröße von Null.

Jedes wiederverwendete Partikel wird leicht gestört. Seine Werte werden durch einen Zufallsgenerator verändert. Der Generator ist eine Normalverteilung, die zufällige Werte um den Nullwerte erzeugt, die auf die Partikel addiert werden. Dadurch werden diese im Raum leicht versetzt und bekommen die Chance, sich auf das Maximum der Wahrscheinlichkeitsverteilung im Zustandsraum zuzubewegen. Im ungünstigen Fall entfernt sich das Partikel jedoch vom Maximum, was jedoch durch weiteres Resampling wieder ausgeglichen wird.

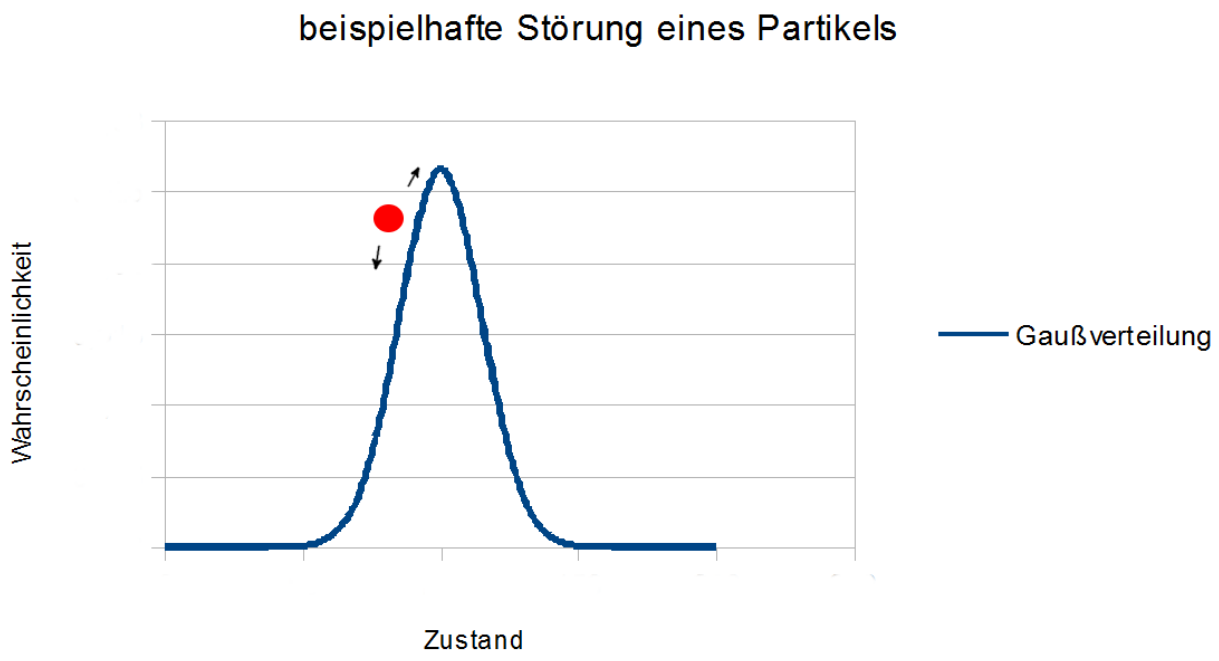

Abbildung 7: Störungen eines Partikels.

Wiederverwendete Partikel werden im Anschluss gestört. Durch leichte Abweichungen der Koordinaten können Partikel dem Maximum der Verteilung entgegen laufen oder aber auch abwandern. Da der Belief in jedem Durchlauf immer wieder auf den neusten Stand gebracht wird, ist es wahrscheinlicher, dass der Partikelcluster durch den besseren Belief gegen das Maximum läuft. Dadurch wird die genaue Position der Messung bestimmt. Ohne diese Störung könnte die gefundene Position sich nicht der Messung nähern.

Ein Teil der Partikel wird immer neu gestreut, um nicht gefundenen Objekten im Raum eine Chance zu geben, doch noch getroffen zu werden.

Das Resampling wird einige Male wiederholt, um die Positionen genauer zu bestimmen. Die Störung der Partikel nimmt über die Iterationen hyperbolisch ab.

Positionsermittlung mit Hilfe von Clustern

Im Anschluss werden aus Laufzeitgründen die besten Partikel mit Hilfe eines sogenannten “online” beziehungsweise sequentiellen k-means zu Clustern zusammen geführt.

Für jedes Partikel wird mit Hilfe eines Abstandsmaßes überprüft, zu welchem Cluster es gehört. Der betrachtete Abstand ist gegenwärtig abhängig von der Varianz des Observers. Kann kein Cluster in der Nähe gefunden werden, wird es selbst zu einem Clusterzentrum. Am Ende sind alle Partikel einem Cluster zugeordnet oder bilden zwangsläufig als einzige einen Cluster. Damit nicht mehrere Cluster nebeneinander existieren, werden in einem weiteren Schritt nahstehende Cluster vereint. Alle Cluster die im Anschluss aus weniger als drei Partikeln bestehen, werden als falsche Positionen kategorisiert und gelöscht.

Die ID von allen Partikel in einem Cluster wird vereinheitlicht und die Partikel ohne Cluster bekommen eine

neue, unbenutzte ID zugewiesen. Sie suchen nach einer ungenutzten ID-Familie im Familienhistogramm und nehmen dann deren Index an.

Evaluierung

Um die korrekte Funktionsweise des Partikelfilters zu testen, wurde zuerst ein künstliches Szenario geschaffen. Auf einem 1000 x 1000 Pixel großen, weißen Bild wurden drei Ellipsen mit bekannten Koordinaten, Winkel, Höhe und Breite eingezeichnet.
Zur Bewertung der Partikel wurde ein Observer basierend auf einer multivariater Gaußverteilung verwendet. Diese nutzt als Dimensionen x- und y-Koordinaten. Da die Winkelerkennung bei den Blobs gegenwärtig nicht einwandfrei funktioniert, wurde vorläufig auf die Orientierung verzichtet.
Es wird eine 2 x 2 Kovarianzmatrix mit Standardparametern erstellt.

$$\begin{pmatrix} Varianz\,fuer\,die\,X-Koordinate & 0 \\ 0 & Varianz\,fuer\,die\,Y-Koordinate \end{pmatrix}$$

Abbildung 8: Kovarianzmatrix für die 2-dimensionale Gaußverteilung. Es sind nur die Varianzen für X und Y eingetragen. Die Kovarianz zwischen X und Y ist nicht von Interesse und wird mit Nullen aufgefüllt.

Nur auf der Diagonalen sind Konstanten eingetragen, da der Abstand des Partikels zum Blob betrachtet wird. Je höher diese Konstanten, desto breiter ist die Gaußverteilung und desto größer ist der Bereich, in dem Partikel gut bewertet werden. Sind die Konstanten klein, wird eine kleinere Fläche gut bewertet.
Wenn der Observer einen Partikel übergeben bekommt, bewertet er diesen bezüglich jedes einzelnen Blobs durch folgende Formel:[7]

$$f_X(x) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left\{ -\tfrac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}.$$

Abbildung 9: Verwendete Formel für die multivariate Normalverteilung zur Bewertung der Partikel.

Der größte, ermittelte Wert wird an den Partikelfilter zurück gegeben. Sind die Werte zu klein oder konnten keine Blobs auf dem Bild gefunden werden, werden diese auf einen Minimal-Belief gesetzt, da immer eine kleine Restwahrscheinlichkeit zurück bleiben soll. Durch Fehler in der Messung bleibt die Möglichkeit offen, dass sich Fische wegen Messfehlern auch an anderen Positionen befinden können. Dieses Vorgehen war ursprünglich als Lösungsansatz für verlorene Blobs gedacht. Unter der Annahme, dass sich geclusterte Partikel mit einer gewissen Mindestwahrscheinlichkeit nicht so schnell auflösen, sollte wiedergefundenen Blobs eine schnelle Zuordnung zu ihrer alten Position ermöglichst werden.
Der laufende Filter wurde mit folgenden Parametereinstellungen getestet.

- Partikelanzahl: 500
- Verhältnis wiederverwendeter Partikel zu neu gestreuten: 85
- Anzahl der Resamplevorgänge pro run-Aufruf: 5
- maximale Partikel pro ID-Familie: 20

- Observer-Varianzen: 30
- Varianz der Störung beim Resampeln: 15

Im Folgenden wurde getestet, wie gut der Partikel statische Ellipsen stabil erkennt. Diese verändern ihre Position nicht und gehen niemals verloren. Der euklidische Abstand zwischen jeder gefundenen Positionen und wahrer Ellipsenposition wurde als Qualitätsmaß verwendet. Die Einheit ist in Pixel. Der Test wendet den Filter immer wieder auf das gleiche Bild an.
Wie man in Abbildung x sieht, findet der Filter nach wenigen anfänglichen Resamplingschritten die Positionen der Ellipsen. Leichte Abweichungen befinden sich zwischen Null und einem Pixel. Es gibt nach 230 Bildern keine verlorenen Objekte oder stark abweichende Positionen.

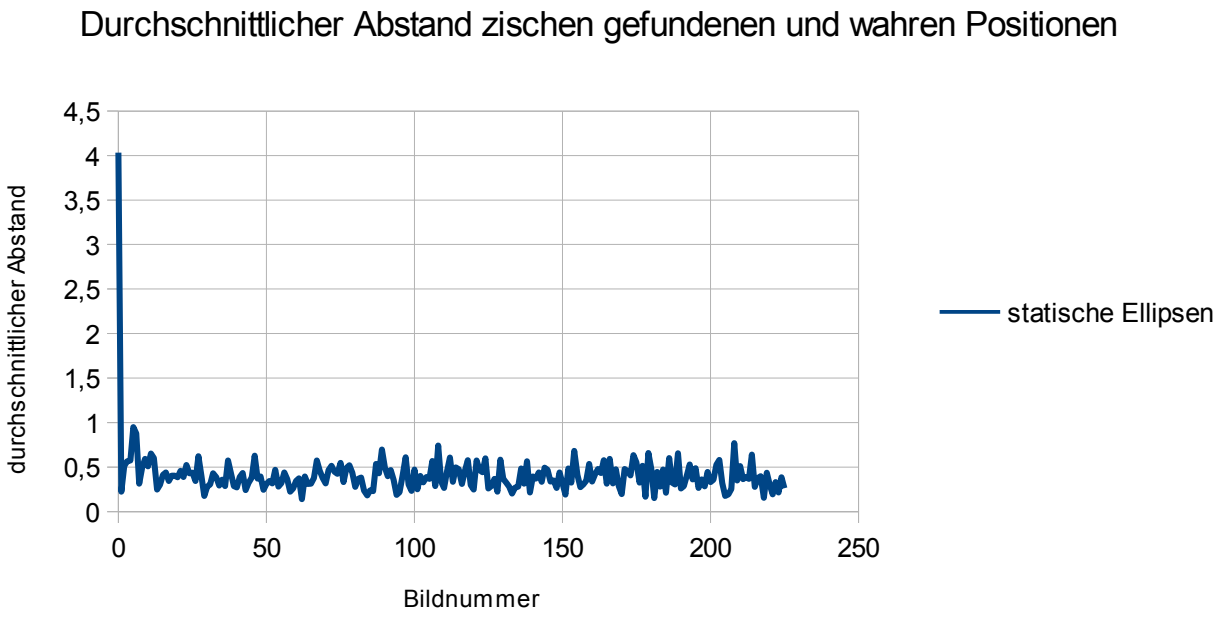

Abbildung 10: Ermittelter Abstand aller gefundener Positionen des Partikelfilters zu den vorgegebenen Ellipsen.

Der euklidische Abstand zwischen allen Positionen des Filters und deren Ellipsen wurde ermittelt und der Durchschnitt gebildet. Man sieht, dass der Filter in den ersten, wenigen Bildern noch keine genauen Positionen ermittelt hat. Hat er jedoch alle Ellipsen richtig erkannt, bleiben diese stabil erhalten. Der euklidische Abstand wird nach 230 Bildern nie größer als Eins. Zwischendurch gingen keine Ellipsen verloren.

Als nächstes wurde getestet, wie sich der Partikelfilter bei einer bewegenden Ellipse verhält. Dafür gab es einen Schieberegler, der die x-Koordinate von einer der drei Ellipsen verändern konnte. Bewegte sich die Ellipse, wurde sie auf dem Bild neu eingezeichnet. Da eine verlorene Position einen unendlichen Abstand zur Ellipse aufweist, wurde diese Distanz mit dem Wert Fünfzig symbolisiert.

Der Partikelfilter war in der Lage, dem Objekt für einhundert Bildern zu folgen. Wenn die Position jedoch verloren ging, gab es Probleme, die Ellipse wieder zu finden. Auch ohne weitere Bewegungen konnte die

Position nicht sofort wieder lokalisiert werden und blieb selbst nach weiteren einhundert Durchläufen noch instabil.

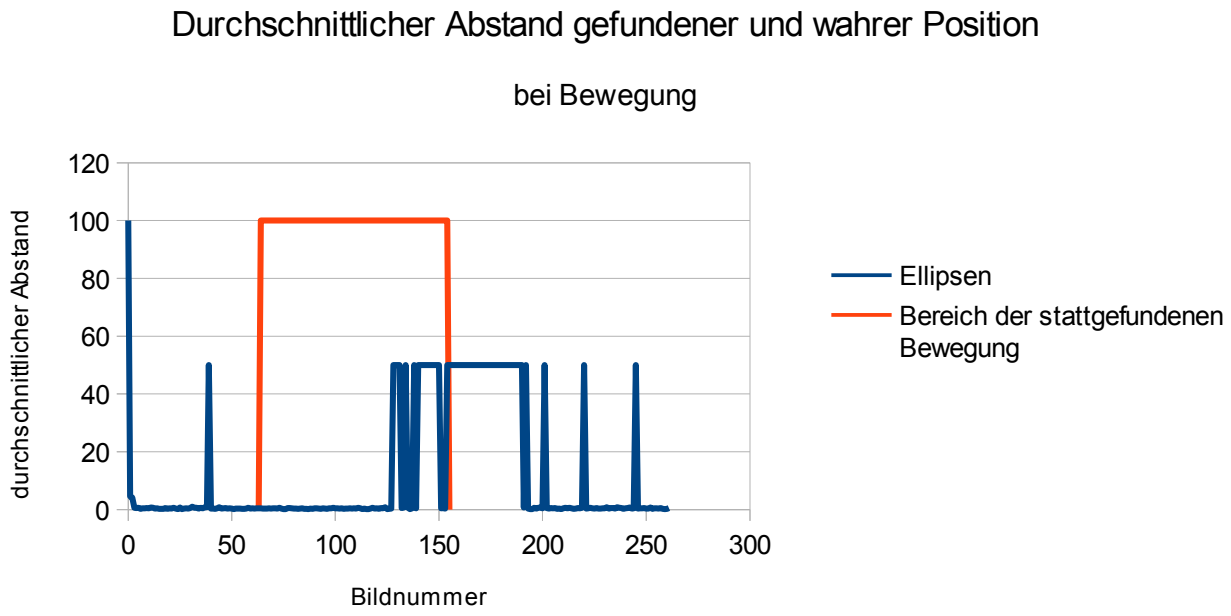

Abbildung 11: Ermittelter Abstand aller gefundener Positionen des Partikelfilters zu den vorgegebenen Ellipsen mit zwischenzeitlicher Bewegung.

Es wurde der euklidische Abstand zwischen Partikelfilter- und Ellipsenpositionen ermittelt. Da verlorene Positionen einen unendlichen Abstand zu ihrer Ellipse aufweisen, wurde der hohe Wert 50 gewählt, um dies zu symbolisieren. Gefundene Ellipsen haben weiterhin eine sehr kleine Abweichung in ihrer ermittelten Position. Dadurch entstehen Peeks, wenn eine Ellipse, typischerweise die bewegte, verloren geht. Ein anfänglicher Peek zeigt, dass auch statische Ellipsen durchaus verloren gehen können. Der rote Datenbereich markiert das Intervall, in der eine Ellipse sich bewegte. Es ist zu sehen, dass anfänglich keine Ortungsprobleme während der Bewegung auftraten. Im letzten Drittel ist die bewegte Ellipse jedoch verloren gegangen und brauchte selbst beim Stillstand einige Zeit, um wieder gefunden zu werden. Stabiles verfolgen kann dadurch unter Umständen problematisch werden.

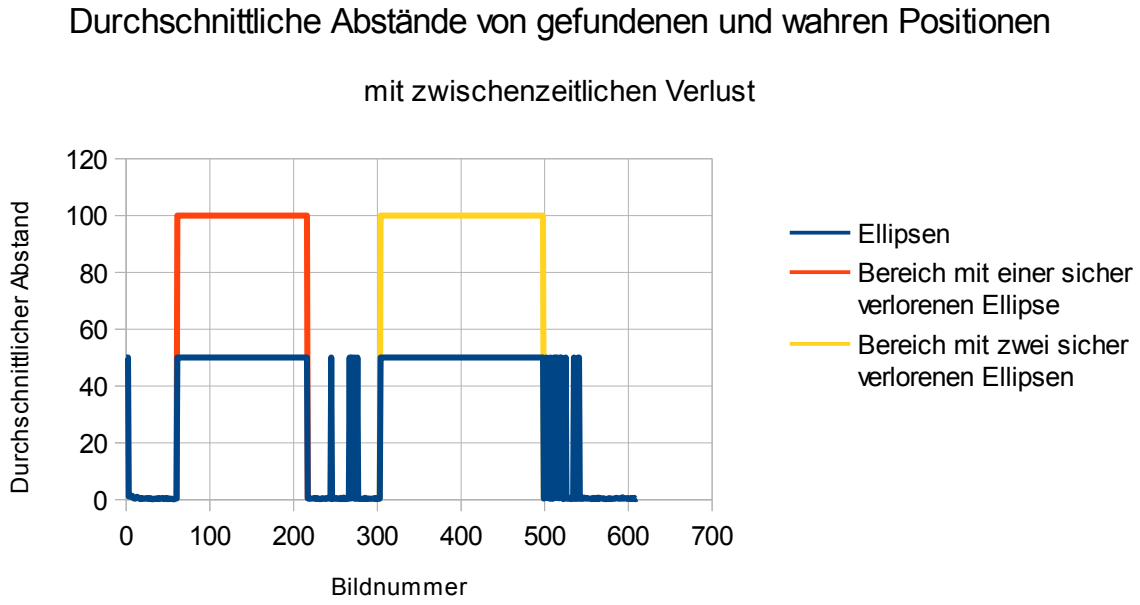

Abbildung 12: Ermittelter Abstand aller gefundener Positionen des Partikelfilters zu den vorgegebenen Ellipsen mit zwischenzeitlich verlorener Ellipse.

Der rote Datenbereich zeigt dass Intervall, in dem eine Ellipse aus dem Bild gelöscht wurde. Bei dem gelben Datenbereich sind es zwei gelöschte Ellipsen. Es ist zu sehen, dass nach einer verlorenen Ellipse, die Position relativ schnell wieder gefunden wurde, jedoch in den nächsten hundert Bildern mit vielen kleinen Peeks noch instabil blieb. Nach zwei verlorenen Ellipsen hatte der Filter mit ungefähr fünfzig Bildern deutlich länger gebraucht, um die Positionen für längere Abschnitte stabil zu finden, hatte aber nachher keine Probleme mit weiteren Peeks.

Ebenfalls wurde getestet, wie sich der Filter verhält, wenn er zwischenzeitlich gar nicht mehr die Möglichkeit bekommt, eine Ellipse zu finden, da sie komplett aus dem Bild verschwindet. Dies ist wichtig bezüglich auf die gegenwärtige Methode der Fischdetektion, da ein Fisch nicht immer als Blob erkannt und vom Observer bewertet werden kann. Der Filter sollte trotzdem in der Lage sein, eine zeitweise verlorene Detektion schnell wieder zu finden. Dafür wurde zunächst eine Ellipse zeitweise aus dem Bild gelöscht. Anschließend eine Zweite. Es ist zu sehen, dass die verlorenen Ellipsen mit anfänglichen Schwierigkeiten wieder gefunden werden. Es braucht dazu zwischen fünfzig und einhundert Bilder.

Trotz dieser Einschränkungen kann der Partikelfilter bei Aufnahmen von Fischen deren Positionen bestimmen und bei Bewegung verfolgen. Im Abbildung 13 sind alle Fische erfolgreich lokalisiert. Dieser Zustand kann über mehrere Bilder stabil bleiben. IDs werden dann korrekt an die Fische vergeben. Allerdings sind Ausfälle von Detektionen keine Seltenheit. In Abbildung 14 ist ein Fische verloren gegangen. Dies reichte aus, um die IDs von mindestens drei weiteren Fischen (5,2,3) zu verändern. Die IDs 1, 0 und 4 sind dabei identisch geblieben.

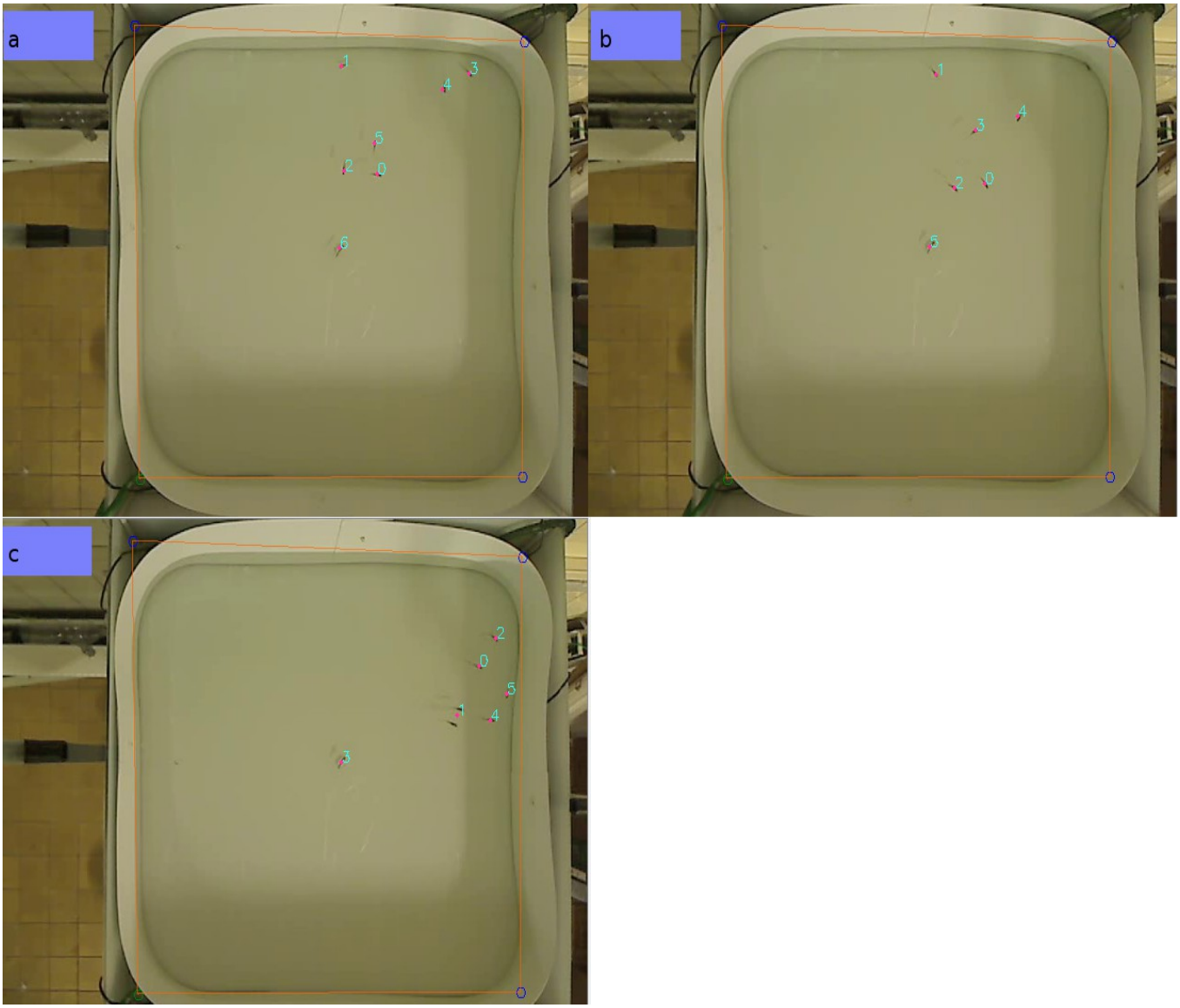

Abbildung 13: Ausgesuchte Bilder eines Fischexperiments. a) Situation, in der alle Fische durch den Partikelfilter gefunden und mit IDs versehen wurden. b) Situation, in der ein Fisch in der rechten, oberen Ecke nicht mehr detektiert wird. Als Konsequenz verändern sich die IDs der restlichen Fische. Vergleiche dazu mit a. c) Situation, bei der zwei Fische zusammen geclustert wurden und die Position, mit der Id 1 versehen, zwischen den Fischen angezeigt wird.

Verlust kann auch erst beim Clustern auftreten, wenn die Varianz des Observers zu hoch eingestellt ist. In Abbildung 15 befindet sich Position 1 genau zwischen zwei Fischen. Dazwischen befindet sich genügend Raum, so dass die Fische nicht zu einem Blob verschmolzen wurden und dies damit kein Fehler der Blobdetektion darstellt.

Diskussion

Der Partikelfilter kann Positionen finden. Er detektiert zuverlässig nur die sichtbar zu suchenden Objekte. Falsche Detektionen im Raum treten nicht auf. Wenn viele Partikel neu initialisiert werden, können sich Cluster außerhalb der Ellipsen kurzfristig bilden. Diese sterben jedoch schnell aus und werden vom Clusterprozess auch nicht als valide erkannt, da nur die besten Partikel verwendet werden, die sich vor allem an den gesuchten Positionen befinden.

Ellipsen, die zeitweise nicht von dem Observer bewertet werden können, werden unterschiedlich schnell wieder gefunden. Durch mehr beziehungsweise gezielteres Neustreuen der Partikel könnte dieser Prozess weiter beschleunigt werden.

Detektionen können verloren gehen, wenn Bewegung auftritt. Die Probleme beim Verfolgen der bewegten Ellipse können durch eine hohe Observer-Varianz und großer Störung wiederverwendeter Partikel verringert werden. Damit lassen sich die recht großen Ellipsen stabil verfolgen.

Problematisch ist es, wenn man kleinere Ellipsen verwendet. Man kann die Varianz nicht zu hoch einstellen, da dann auch die nicht zugehörige Umgebung ebenso gut bewertet wird. Dadurch treten Fehler auf, wenn viele kleine Ellipsen nah zusammen kommen. Die Grenzen zwischen den beiden Objekten verwaschen und können zu einem Cluster verschmelzen. Nimmt man jedoch kleine Varianzen, wird das stabile Verfolgen und Wiederfinden eines verlorenen Objektes erschwert. Dieses Szenario tritt häufig bei den Fischen im Wassertank auf. Die Guppys sind relativ klein und schwimmen bevorzugt in engen Schwärmen.

Bei Experimenten gehen mitunter deshalb gefundene Fische zu häufig verloren. Einerseits ist das verschuldet durch den Blob-Observer, der die Partikel nur korrekt bewerten kann, wenn er wirklich alle Fische als Blobs übergeben bekommt. Andererseits scheint es beim Resampeln das Problem zu geben, dass einige Fische trotz guter Partikel selten oder gar nicht wiederverwendet werden. Andere Fische ziehen die Partikel an sich. Ausgelöst wird dieses Fehlverhalten nach bisherigen Kenntnisstand durch einen Fehler bei den ID-Familien. Wenn sich eine Familie bei einem Ort gesammelt und das Maximum erreicht hat, wird sie korrekterweise nicht weiter verwendet. Liegt jedoch ein weiteres, gutes Partikel mit einer anderen ID in der Nähe, entsteht am gleichen Ort eine zweite, unabhängige Familie. Dadurch kann ein Cluster mehr Partikel besitzen als es durch die ID-Familie erlaubt ist. Ein erster Lösungsansatz wurde beim Clustern umgesetzt. Alle Mitglieder eines Clusters sollen dieselbe ID bekommen. Dementsprechend werden mehrere Familien zu einer vereint. Da durch das Streuen jedoch immer wieder neue Partikel mit anderen IDs in diesen Clustern auftreten können, taucht diese Problem über die Zeit immer wieder auf.

Die Cluster-ID wird in jedem Bild neu über eine einfache Schleife vergeben, weswegen die korrekte ID der Fische bei Störungen nicht zugeordnet werden kann. Es kommt zu Clusterungenauigkeiten, wenn zwei Positionen nah beieinander liegen oder Cluster gar ganz verloren gehen. Die IDs springen von Fisch zu Fisch. Das Phänomen tritt ebenfalls bei verlorenen Blobs auf. Es ist noch nicht implementiert, wie der Clusterpro-

zess des Partikelfilters mit verlorenen Blobs umgehen soll. Das Clustern kann außerdem nicht mit Blobs umgehen, die sich kurzfristig vereinen.

Da die Laufzeit hoch ist, kommt das Live-Tracking ins Stocken. Das Video hat beim Nearest Neighbours mindestens 20 FPS. Abhängig von der Anzahl der vorhandenen Partikel, schwankt die Leistung des Filters zwischen ein und elf FPS. Es muss dabei angemerkt werden, dass weniger Partikel den Vorgang zwar beschleunigen, aber zugleich die Genauigkeit der Positionen mindert. Ein großer Teil der Laufzeit geht bei dem Observer, dem Resampeln und des Clusterns verloren. Laut dem Profiler von VisualStudio 2012 teilen sie sich diese Komponenten jeweils ein Drittel der Laufzeit.

Ausblick

Obwohl es Ideen zur Verbesserung gab, konnten die zuvor beschriebenen Probleme der Implementierung nicht mehr im zeitlichen Rahmen gelöst werden.

Wenn es einen effizienteren und genaueren Observer gäbe, könnte man gleich zwei Baustellen verbessern: Der Filter sollte einiges an Laufzeit einsparen können und hätte mit weniger Ungenauigkeiten im System zu kämpfen. Abgesehen von verlorenen Blobs könnte man mit überlappenden Blobs detailierter umgehen. Im Gespräch steht ein Observer, der sich auf die Konturen der Fische stützt. Ein Fisch hat eine ellipsoide Form mit einer bestimmten Größe. Verändert sich die Größe oder gar die ganze Form, wüsste man, dass es sich nicht mehr nur um einen einzelnen Fisch handelt.

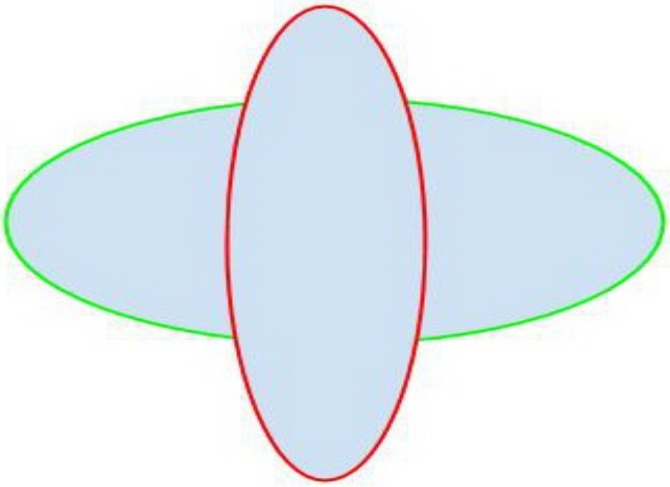

Abbildung 14: ´Zwei sich schneidende Ellipsen.

Beim Blobdetektor und Blob-Observer wurde diese Form als ein einziger Blob beziehungsweise als Position erkannt. Eine Analyse der Kontur könnte vorhersagen, dass es sich nicht um eine Ellipse sondern höchstwahrscheinlich um zwei handeln. Wenn zusätzlich ein Richtungswinkel bestimmt werden kann, ist es möglich beim Auftreffen und Auseinanderstoßen klare Aussagen über das einzelne Objekt und seiner ID zu treffen.

Auch sollte im Observer wie auch beim Clustering die Höhe und Breite beachtet werden, damit sich die Partikel nicht kreisförmig sonder mehr ellipsoid ansammeln.

Durch die hierarchische Clusterstruktur ist das Clustern ineffizient und ungenau. Das Distanzmaß könnte neu überarbeitet werden, damit nicht mehrere eng benachbarte Cluster entstehen. Da Partikel bewegenden Zielpositionen hinterher wandern, sollten die Familien den neuen Clustern womöglich zugeordnet werden. Bei den ID-Familien der Partikel bedarf es eine Überarbeitung, damit nicht mehrere Familien nebeneinander entstehen und die lokale Einschränkung des Ansammelns zu vieler Partikel an einer Position teilweise umgangen wird. Möglicherweise könnte ein Ähnlichkeitsmaß nah stehende Partikel mit derselben ID versehen.

Quellenverzeichnis

[1] Klassische Verhaltensbiologie, http://www.verhaltenswissenschaft.de/Verhaltensbiologie/verhaltensbiologie.htm, aufgerufen am 30.4.2014

[2] Theoretische Biologie, http://www.spektrum.de/lexikon/biologie/theoretische-biologie/66247, aufgerufen am 30.4.2014

[3] ICondensation: Unifying low-level and high-level tracking in a stochastic framework, Michael Isard and Andrew Blake, Proc 5th European Conf. Computer Vision, Vol. 1 893-908, 1998, aufgerufen am 24.4.2014

[4] Bildverarbeitung und Algorithmen, WS07 5.1 ©Konen, Zielke, http://www.gm.fh-koeln.de/~konen/WPF-BV/BV05.PDF , aufgerufen am 28.4.2014

[5] Swain, Daniel T., Iain D. Couzin, and Naomi Ehrich Leonard. "Real-time feedback-controlled robotic fish for behavioral experiments with fish schools."*Proceedings of the IEEE* 100.1 (2012): 150-163.

[6] Bonnet, Frank, et al. "Development of a mobile robot to study the collective behavior of zebrafish." *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*. IEEE, 2012.

[7] Hartung, Joachim| Elpelt, Bärbel, Multivariate Statistik, Lehr- und Handbuch der angewandten Statistik, 7. Auflage, München Oldenbourg Wissenschaftsverlag, 2007

Vorarbeiten

Rami Akkad, Entwicklung eines visuellen Trackingsystems für biomimetische {Roboterfische}, Freie Universität Berlin, Diplomarbeit, 2012,

Stefan Forgo, Detektion und Tracking von Individuen eines Fisch-Schwarms und Extraktion von Verhaltensmustern, Freie Universität Berlin, Bachelorarbeit, 2013

Viet Hai Nguyen, Development of an omni-directional mobile robot platform for research on collective behaviour in fish shoals, Freie Universität Berlin, Masterarbeit, 2013

Verwendete Software Dritter

Microsoft VisualStudio2012

Bradski, Gary. "The opencv library." *Doctor Dobbs Journal* 25.11 (2000): 120-126.

Qt Proect, http://qt-project.org/, aufgerufen am 1.5.2014

Bildverzeichnis

Abb.1 Versuchsaufbau, erstellt mit SketchUp 2014

Abb.2 Bilder aus einem aufgenommenen Experiment an der HU Berlin, erstellt mit Microsoft Snipping Tool

Abb.3 Implementierungsübersicht, erstellt mit Google Docs

Abb.4 Abhängigkeit des Beliefs, erstellt mit Google Docs

Abb.5 Kumulative Summe, erstellt mit Google Docs

Abb.6 Familienhistogramm, erstellt mit OpenOffice Calc

Abb.7 Störung eines Partikels, erstellt mit OpenOffice Calc, bearbeitet mit Gimp

Abb.8 Kovarianzmatrix, erstellt mit http://www.codecogs.com/latex/eqneditor.php

Abb.9 multivariate Normalverteilung, Quelle: http://upload.wikimedia.org/math/0/7/c/07c2e5d338738ec96ad9a63d0f43bb95.png, , aufgerufen am 17.4.2014

Abb.10 Statische Ellipse, erstellt mit OpenOffice Calc

Abb.11 Bewegte Ellipse, erstellt mit OpenOffice Calc

Abb.12 Verschwundene Ellipse, erstellt mit OpenOffice Calc

Abb.13 Bildbeispiele vom Partikelfilter aus einem aufgenommenen Experiment an der HU Berlin, erstellt mit Microsoft Snipping Tool

Abb.14 Konturbeispiel, erstellt mit Google Docs