TECHNISCHE UNIVERSITÄT MÜNCHEN

LEHRSTUHL FÜR DATENVERARBEITUNG

TECHNISCHER REPORT

Realisierung eins Self-Splitting PSO Algorithmus für Multiple Object Tracking

Alexander Kronthaler,
Fakheredine Keyrouz,
Klaus Diepold

30. Januar 2009

Kurzfassung

Im Rahmen dieser Arbeit wurde der Partikelschwarmalgorithmus in Bezug auf Objektnachführung in Videosequenzen als praktische Anwendung erforscht.

Zunächst wurden dazu die Grundlagen der Particle Swarm Optimization (PSO) und der gegenwärtige Stand der Forschung im Bereich Tracking erarbeitet. Als wichtigste Methoden werden kurz die Hidden-Markov-Modelle, der Optische Fluss, Blockmatching und andere Verfahren dargestellt.

Zu Forschungszwecken wurde eine PSO-basierende Tracking-Software erstellt, die den aktuellen Stand der Technik widerspiegelt. Hierbei wurden die Auswirkungen bestimmter Parameteränderungen, verschiedene Varianten der PSO Gleichungen und Einflussbereiche und Verhaltensweisen des Schwarms erforscht. Bei diesen Untersuchungen zeigte sich, dass der Schwarm des PSO-Algorithmus bestimmte Verhaltensweisen zeigt, die sich zur Realisierung neuer, bisher nicht erforschter Möglichkeiten eignet.

Diesen neuen Ideen lag, wie bereits dem PSO-Konzept nach Kennedy und Eberhart, eine Inspiration aus der Natur zugrunde: Wenn es möglich ist, mit einem Schwarm ein bestimmtes Ziel zu suchen, muss es auch möglich sein, mit mehreren, voneinander autonomen Schwärmen mehrere Ziele gleichzeitig zu suchen. Schließlich existieren in der freien Natur eine Vielzahl verschiedener Vogelarten, die in einem Schwarm unterwegs sind, jedoch nicht miteinander kommunizieren und verschiedene Ziele anfliegen. Etwa kommunizieren Schwalben nicht mit Raben und umgekehrt.

Auf dieser Grundlage konnte ein Self-Splitting-PSO Algorithmus realisiert werden, der mit n autonomen Schwärmen n unabhängige Objekte in einer Videosequenz nachführen kann. Bisher existierte nur das Konzept der hierarchischen Schwärme, die mit mehreren voneinander abhängigen Schwarmgruppen voneinander abhängige Objekte verfolgen können. Außerdem ist es bei dem neu entstandenen Self-Splitting PSO möglich, dass mehrere Objekte bei räumlicher Nähe zu einem Objekt "'verschmelzen"' oder dass aus einem Objekt wieder zwei autonome Objekte entstehen.

Um mit der neuen Software für verschiedene Situationen verschiedene Kriterien verwenden zu können, wurde eine adaptive Fitnessfunktion programmiert, die den autonomen Schwärmen des PSO die Qualität einzelner Positionen anzeigt und sich an wechselnde Bildverhältnisse anpassen kann. So können auch komplexe Situationen wie sehr schnelle Bewegungen oder starke Lichtwechsel erfolgreich nachgeführt werden.

In einem Vergleichsverfahren wurde schließlich die Anwendbarkeit der neuen Idee verifiziert. Damit erhält die Idee des PSO, die bisher noch nicht an die Ergebnisse mit Hidden-Markov-Modellen im Bereich Tracking heranreichen konnte, einen neuen Impuls.

Contents

List of Figures

List of Tables

Chapter 1

Einleitung

Seit Jahrzehnten gibt es in den Ingenieurwissenschaften mathematische Probleme, die nicht oder nur mit sehr hohem Aufwand zu lösen sind. Aus diesem Grund entstand die numerische Mathematik, die versucht, Algorithmen zur Berechnung dieser wiederkehrenden Probleme zu konstruieren und zu analysieren. Hierbei handelt es sich oft um Optimierungsprobleme: Anstelle des mathematischen Grundsatzes, dass ein Ergebnis entweder falsch oder richtig sein kann, tritt eine oft komplexe Fitness- oder Zielfunktion, die das gefundene Ergebnis einordnet. Bei einem eigentlichen Optimierungsproblem wird nur der beste Wert, der in einer bestimmten, endlichen Zeit gefunden werden kann, als Ergebnis definiert. Man will also zumindest eine möglichst gute Lösung finden, falls das Ermitteln einer optimalen Lösung nicht praktikabel ist. Diesen Vorgang bezeichnet man als Approximation. Von dieser Methodik gibt es jedoch zahlreiche Abwandlungen. Hierzu zählen beispielsweise Entscheidungsprobleme, bei denen ein Ergebnis ab einem bestimmten Schwellwert per Definition als richtig oder passend zu werten ist.

Ein wichtiges Anwendungsgebiet dieser Optimierungsprobleme ist das Lokalisieren von bewegten Objekten in einem Bild und das Verfolgen beziehungsweise Nachführen dieser Objekte in Videosequenzen. Dieser als Tracking bezeichnete Vorgang stellt stets ein Optimierungsproblem dar: Anhand von Informationen wie Farbe oder Form eines bestimmten Objektes wird ein Muster oder eine Maske im Bild gesucht. Da sich Objekte nun durch Faktoren wie Lichtverhältnisse, Bewegungen oder Verschiebungen ändern, muss nach einem Muster gesucht werden, das dem zuvor gefundenen Muster möglichst nahe kommt. Dies wird durch verschiedene Verfahren bewerkstelligt. Das sogenannte Blockmatching Verfahren sucht etwa stets nach einem Pixelblock einer festen Größe. Ob

der gefundene Block nun dem gewünschten Block ähnlich ist, wird durch Verfahren wie etwa SAD (Sum of Absolute Differences) oder MSE (Mean Squared Error) festgestellt. Um nicht zufällig Blöcke überprüfen zu müssen gibt es zusätzliche Verfahren wie etwa den Three Step Search, der möglichst immer in der Richtung weiterwandert, in der der geringste Fehler gefunden wurde. Die verschiedenen Ausprägungen des Optischen Flusses versuchen dagegen, anhand der Verschiebung einzelner Teilblöcke auf die Bewegung des Objektes zu schließen. Bewegungsvektoren können zusätzlich die Verschiebung der einzelnen Blöcke anzeigen. Hidden Markov Modelle (HMM) benötigen umfangreiches Trainingsmaterial um zu sinnvollen Ergebnissen zu gelangen. Anhand dieses Trainingsmaterials werden dann ebenfalls Blöcke oder Bildteile gesucht, die sich ähnlich verhalten.

Die meisten dieser Verfahren arbeiten mit statischen Kriterien. Damit können zahlreiche Situationen jedoch nicht hinreichend bewältigt werden, etwa bei schnellen Bewegungsänderungen oder starker Änderung des Lichteinfalls sind diese problematisch. Zusätzlich hat jedes dieser Verfahren das Problem, dass alle Blöcke nacheinander abgearbeitet werden.

Ein relativ neues Verfahren zur Lösung von Optimierungsproblemen wurde 1995 von J. Kennedy und R.C. Eberhart entwickelt. Der Ansatz der Particle Swarm Optimization hat seine Wurzeln im Sozialverhalten von Fisch- oder Vogelschwärmen. Um bestimmte Strecken zurückzulegen oder Nahrung zu finden bewegen sich diese Schwärme nach bestimmten Mustern. Diese Muster können durch Algorithmen angenähert und reproduziert werden, 20 oder mehr Mitglieder des Schwarms untersuchen dabei gleichzeitig jeweils einen anderen Bereich. Jedes Mitglied des Schwarms liefert in jedem Zeitschritt die Performanz eines durch die Fitnessfunktion bewerteten Bereiches. Anschließend richten sich alle Partikel mit neuer Geschwindigkeit in eine neue Orientierung aus, die verstärkt in Richtung der performanten Bereiche liegt. So wird sich dem Optimum schrittweise angenähert bis nach endlicher Zeit kein besserer Bereich mehr gefunden werden kann und eine vordefinierte Abbruchbedingung ausgelöst wird.

In jüngster Zeit zeigen diese Partikelschwarmalgorithmen in zahlreichen Forschungsergebnissen eine bessere, schnellere und vor allem robustere Nachführung von bewegten Objekten als herkömmliche Vorgehensweisen. Eines der Hauptprobleme des PSO besteht jedoch darin, dass stets nur ein Schwarm zum Verfolgen eines Objektes verwendet wird. Ist beispielsweise bei einer Überwachungssoftware gewünscht, dass

zwei oder mehr Personen gleichzeitig nachgeführt werden sollen, ist PSO bis dato nicht verwendbar. Mit anderen Methoden, z.B. mit Hidden Markov Modellen, ist diese Problematik längst gelöst. Wäre PSO fähig, diese Problemstellung zu bewältigen, wäre es anderen Methoden durch seine hohe Robustheit bei Farb- oder Helligkeitsänderungen unter Umständen sehr weit voraus.

Dieser Thematik widmet sich diese Arbeit. Zunächst wurden dazu verschiedene Ansätze und Varianten des PSO Algorithmus untersucht und dann sukzessive weiterentwickelt. Zur Überprüfung und Verifizierung der entstandenen Algorithmen erfolgten Implementierungen in Matlab. Die Grundidee des Partikelschwarmalgorithmus wurde verändert und ist nun zum Tracking von n sich voneinander unabhängig bewegenden Objekten nutzbar. Dies wird durch n voneinander völlig unabhängige Schwärme erreicht, die jeweils ein durch die Fitnessfunktion vorgegebenes Ziel verfolgen. Weiterhin ist es nun erstmals gelungen, mit Partikelschwarmalgorithmen sogenannte Self-Splitting Komponenten zu implementieren: Verschwinden etwa Objekte hintereinander oder überkreuzen sie sich, erkennt sie der Algorithmus als ein zusammenhängendes Objekt. Vergrößert sich der Abstand wieder, werden sie erneut als zwei eigenständige Objekte verfolgt.

Um die Berechnungszeit der Ergebnisse niedrig zu halten und das Konzept für Echtzeitanwendungen im Bereich Bildverarbeitung verwendbar zu machen wurde eine spezielle adaptive Fitnessfunktion geschaffen, die während des Trackingvorganges verschiedenste Kriterien benutzen kann. So werden in bestimmten Situationen, die als unkritisch eingestuft werden, nur wenige Kriterien verwendet, während in komplexeren Situationen weitere Entscheidungsfaktoren hinzugezogen werden.

Während der Entstehung dieses neuartigen Self-Splitting PSO mit adaptiver Fitnessfunktion wurden umfangreiche Vergleichstests mit anderen gängigen Methoden durchgeführt: Vor allem die hohe Robustheit in komplexeren Situationen als auch die Flexibilität durch die adaptive Fitnessfunktion ist hier hervorzuheben.

Chapter 2

Grundlagen

2.1 Der Partikelschwarm-Algorithmus

2.1.1 Definition

Die Idee der Particle Swarm Optimization (PSO) wurde erstmals 1995 von Kennedy/Eberhart erdacht und publiziert [1]. Nach den Erfindern lag der Idee der Versuch zugrunde, erfolgreiche Konzepte aus der Natur zu kopieren. Diese Ansätze werden unter dem Sammelbegriff "'Artificial Life"' zusammengefasst, der sich der Lösung folgender zwei Probleme widmet [2]:

1. "'Artificial Life"' befasst sich damit, wie rechenintensive Verfahren zur Lösung biologischer Phänomene benutzt werden können.

2. "'Artificial Life"' befasst sich damit, wie biologische Techniken für rechenintensive Probleme verwendet werden können.

Bei PSO handelt es sich um den letztgenannten Fall: Ein biologisches Suchverfahren, das von verschiedenen Tierarten intuitiv verwendet wird, soll bei der Lösung von mathematischen Optimierungsproblemen helfen, bei denen andere Verfahren nicht oder nur in zu langer Zeit das gewünschte Ergebnis liefern können. Der Versuch, sich bei schwierigen mathematischen Problemen mit biologischen Techniken helfen zu lassen, ist nicht neu: Neuronale Netze befassen sich seit geraumer Zeit mit dem Nachbilden eines vereinfachten Modells des menschlichen Gehirns, genetische Algorithmen versuchen, die Entwicklung der menschlichen Evolution als Entscheidungssystem für rechenintensive Probleme zu benutzen [2]. Das erstgenannte Verfahren leidet allerdings unter der Tatsache, dass das menschliche Gehirn eine enorme Trainingszeit benötigt,

um komplexe Problemstellungen lösen zu können. Zwar kann diese Zeit mit heutigen Rechnern stark verringert werden, trotzdem ist eine Lösung von Problemen mit wechselnden Bedingungen und Anforderungen nach wie vor höchst komplex. Schließlich beträgt die Zeitspanne, die der Mensch benötigt, um sich halbwegs in der von ihm geschaffenen Welt zu behaupten, nahezu 20 Jahre. Das zweitgenannte Verfahren, die genetischen Algorithmen, bauen ausschließlich auf Verhaltensmuster und Urbilder, die dem Individuum durch Vererbung in eine neue Generation mitgegeben werden. Diese Urbilder werden nach dem Schweizer Psychiater und Psychologen Carl Gustav Jung als Archetypen bezeichnet. Archetypen beruhen auf einer Instinktgrundlage und stellen eine Art von "'arttypischen Programmen"' dar. Sie haben sich evolutionär entwickelt, in dem Sinne, dass instinktives Verhalten die Kultur und Bewusstseinsentwicklung des Menschen prägte und dass bestimmte psychische Strukturelemente für das Überleben der Art von Vorteil waren, die dann als archetypische Strukturen über Jahrtausende sich entwickelten und vererbt wurden [3]. Beispiel für eine solche Grundstruktur ist das Aussehen eines Bären, das jedem Menschen von der Grundform her vertraut ist und in früheren Zeiten offenbar lebensnotwendig war. Diese Grundstrukturen sind jedoch nur ein winziger Ausschnitt von dem, was ein Mensch in seinem Leben erlernen kann. Die genetischen Algorithmen berücksichtigen jedoch nur diesen winzigen Ausschnitt.

Die schwarmbasierenden Methoden versuchen, das notwendige Training, das vor allem bei den neuronalen Netzen aufwändig ist, während des Suchprozesses zu bewältigen. So sind Echtzeitanwendungen mit wechselnden Anforderungen wesentlich leichter realisierbar. Die zwei populärsten Vertreter der Partikelschwarmalgorithmen sind die Ant Colony Optimization (ACO) und die Particle Swarm Optimization (PSO). ACO optimiert dabei den kürzesten Weg zwischen zwei oder mehr Stationen, wie es etwa beim "'Travelling Salesman Problem"' (eine bestimmte Anzahl Stationen muss jeweils einmal besucht worden sein, der insgesamt kürzeste Weg soll verwendet werden) nötig ist [4]. Analog zur Natur senden dazu einzelne "'Ameisen"' Pheromone (Duftstoffe) aus, die Informationen über Wegnetze enthalten. Die kürzeren Wege enthalten nach einiger Zeit höhere Pheromonkonzentrationen als andere.

PSO beschäftigt sich dagegen damit, möglichst schnell eine "'beste"' Position innerhalb eines n-dimensionalen Suchraums zu finden, wobei die Qualität der einzelnen Positionen eindeutig durch eine Fitnessfunktion beschreibbar sein muss. Die Idee zu diesem Algorithmus lieferte das Sozialverhalten von Fisch- und Vogelschwärmen.

2.1.2 Grundlagen

Bei Fisch- und Vogelschwärmen, die in der freien Natur leben, sind spezielle Verhaltensmuster aufzufinden, die bei der Nahrungssuche oder bei Gefahr angewendet werden können. In diesen Situationen wird nach Kennedy/Eberhart durch zwei verschiedene psychologische Komponenten das Handeln jedes einzelnen Individuums beeinflusst, einer individuellen und einer sozialen Komponente [5]. Die individuelle Komponente steht dabei für das Konzentrieren auf eigene Erfahrungen. Das Individuum achtet dann weniger auf das kollektive Verhalten des Schwarms und verlässt sich mehr auf sich selbst. Die soziale Komponente rückt dagegen das Verhalten des gesamten Schwarms in den Vordergrund: Handelt beispielsweise der ganze Schwarm oder ein großer Teil davon nach einem bestimmten Muster, ist die Wahrscheinlichkeit groß, dass das Individuum diesem Handlungsmuster folgt. Kennedy/Eberhart gehen von der Annahme aus, dass sich diese beiden Komponenten in der Natur selbstständig immer wieder neu gewichten, d.h. in bestimmten Situationen kann die soziale Komponente, in bestimmten die individuelle Komponente stärker gewichtet sein. Deshalb wird diese Gewichtung mit Zufallszahlen versehen [5], da sie keinem erkennbaren Muster folgt.

Die beiden grundständigen Formeln des PSO befassen sich mit zwei Fragen: Mit welcher Geschwindigkeit fliegt das Individuum in welcher Richtung in der nächsten Runde? An welcher Position befindet es sich dann in dieser nächsten Runde? Der gesamte Suchprozess findet also rundenbasiert statt, diese zwei Fragen werden iterativ in jeder Suchrunde neu durchlaufen. Die Anzahl der Suchrunden kann dabei bis zu mehreren Tausend betragen. Die beiden grundlegenden Formeln des PSO lauten:

$$\begin{aligned} v_{x,y}(t) = \omega \cdot v_{x,y}(t-1) + c_1 \cdot \phi_1 \cdot (pbest_{x,y} - p_{x,y}(t-1)) \\ + c_2 \cdot \phi_2 \cdot (gbest - p_{x,y}(t-1)) \end{aligned} \tag{2.1}$$

$$p_{x,y}(t) = p_{x,y}(t-1) + v_{x,y}(t) \tag{2.2}$$

Formel 2.1 zeigt das in jeder Runde stattfindende Erneuern der Geschwindigkeit eines jeden Individuums: Die Geschwindigkeit in Richtung x und y zum Zeitpunkt t. ω, c_1 und c_2 sind dabei vordefinierte Gewichtungsfaktoren, die die Gewichtung der individuellen und sozialen Komponente neben den Zufallszahlen ϕ_1 und ϕ_2 mitbestimmen. Standardmäßig werden diese beiden Gewichtungsfaktoren gleich stark gewichtet,

etwa mit dem Faktor 2 oder 2.05 [6]. In der Auslegung dieser Faktoren gibt es jedoch zahlreiche Abweichungen, je nachdem für welche Anwendung und unter welchen Bedingungen der PSO verwendet wird.

Um nun die neue Geschwindigkeit zum Zeitpunkt t zu berechnen, wird die alte Geschwindigkeit verwendet. Bei $pbest$ handelt es sich um die beste Position, die das entsprechende Individuum in dieser Runde gefunden hat, $gbest$ stellt das aktuelle globale Optimum des gesamten Schwarms dar, also die Position, die das erfolgreichste Individuum in dieser Suchrunde gefunden hat. Dieses erfolgreichste Ergebnis mit Position ist dabei ständig allen Mitgliedern des Schwarms bekannt, sie können also untereinander kommunizieren. Die Gewichtung der sozialen Komponente bestimmt nun, wie schnell und in welcher Richtung sich ein jedes Individuum dieser in der aktuellen Runde bester Position annähert.

Gleichung 2.1 und 2.2 stellen also die Idee des PSO klar heraus: Es soll für jedes Individuum des Schwarms zu jedem neuen diskreten Zeitschritt t eine neue Geschwindigkeit mit Richtungsangabe und daraus abgeleitet die neue Position berechnet werden. Da mit mehreren Zufallszahlen und sich mit der Zeit verändernden Gewichtungsfaktoren gearbeitet wird, handelt es sich um ein stochastisches Verfahren. Es ist möglich aber nicht zwingend so, dass der Algorithmus in zwei Durchläufen mit exakt denselben Bedingungen genau dasselbe Verhalten aufweist.

2.1.3 Varianten des PSO

Die beschriebene Form des PSO stellt die Grundform des PSO dar. Nach der Erfindung des Partikelschwarmalgorithmus entstanden jedoch zahlreiche Varianten, die eine Verbesserung der Performanz des PSO zum Ziel hatten. Die wichtigsten seien hier kurz vorgestellt.

Hier sind etwa die Forschungsarbeiten von Trelea zu nennen [7], in denen die Suche nach einem optimalen Ergebnis durch Variation der Parameter noch schneller erfolgreich beendet werden soll. Trelea verwendet dafür verschiedene Parametersätze, in denen er den individuellen und den sozialen Teil des Individuums unterschiedlich stark gewichtet. Zwar werden die beiden Komponenten durch zwei Zufallszahlen auch nach Kennedy/Eberhart unterschiedlich gewichtet, jedoch nicht nach einem festen Schema. Der Parametersatz 1 nach Trelea lautet:

$$v_{x,y}(t) = 0.729 \cdot v_{x,y}(t-1) + 1.494 \cdot \phi_1 \cdot (pbest_{x,y} - p_{x,y}(t-1)) \\ +1.494 \cdot \phi_2 \cdot (gbest - p_{x,y}(t-1)) \tag{2.3}$$

Der Faktor ω wird nach Trelea also durch den festen Wert 0.729 ersetzt. Das hat zur Folge, dass dieser Wert während allen Iterationen dieser Suchrunde gleich bleibt. Nach Kennedy/Eberhart verändert sich dieser Wert über der Zeit: Von relativ hohen Anfangswerten zu Beginn fällt dieser Wert, bis er ab einem bestimmten Zeitpunkt konstant bleibt. Im original PSO wird der vorherigen Geschwindigkeit also ein variabler Wert zugewiesen, während er nach Trelea fest ist. Die beiden Gewichtungsfaktoren, nach Kennedy/Eberhart entweder variabel oder abhängig von der Anwendung [5], sind nach Trelea ebenfalls fest. Der zweite Parametersatz nach Trelea lautet:

$$v_{x,y}(t) = 0.6 \cdot v_{x,y}(t-1) + 1.7 \cdot \phi_1 \cdot (pbest_{x,y} - p_{x,y}(t-1)) \\ +1.7 \cdot \phi_2 \cdot (gbest - p_{x,y}(t-1)) \tag{2.4}$$

Bis auf veränderte Zahlenwerte bleibt der Grundansatz dabei gleich. Nach Kennedy/Eberhart wird ω berechnet nach der Formel

$$\omega(i) = \frac{iw_2 - iw_1}{iw_e - 1} \cdot (i-1) + iw_1 \tag{2.5}$$

iw_1 und iw_2 sind dabei der Start- und Endpunkt für den Gewichtungsfaktor, iw_e ist der Finalwert für den Gewichtungsfaktor, der dann nicht mehr unterschritten wird.

Maurice Clerk verwendet dagegen einen Beschränkungsfaktor χ, der mit dem gesamten Term multipliziert wird. Die Formel lautet dann

$$v_{x,y}(t) = \chi \cdot (\omega \cdot v_{x,y}(t-1) + c_1 \cdot \phi_1 \cdot (pbest_{x,y} - p_{x,y}(t-1)) \\ +c_2 \cdot \phi_2 \cdot (gbest - p_{x,y}(t-1))) \tag{2.6}$$

Der Wert von χ ist dabei abhängig von c_1 und c_2. Sind diese beiden Faktoren zusammenaddiert kleiner oder gleich 4, ist $\chi = 1$ und damit vernachlässigbar. Ansonsten gilt

$$\psi = c_1 + c_2 \tag{2.7}$$

$$\chi = \frac{2}{\left|(2 - \psi - \sqrt{(\psi^2 - 4 \cdot \psi)})\right|} \tag{2.8}$$

Diese Abweichungen vom Originalkonstrukt sind jedoch alle umstritten. Zwar können sie unter bestimmten Bedingungen vorteilhaft sein, eine generelle Verbesserung konnte bislang jedoch nicht bewiesen werden. Deswegen wird gegenwärtig auch meistens das Originalkonstrukt verwendet, etwa in [8].

2.1.4 Parameter des PSO

Abgesehen von den Basisgleichungen 2.1 und 2.2 gibt es noch zahlreiche weitere Parameter, die für den PSO definiert werden müssen. Diese sind oft variabel und müssen je nach gewünschter Anwendung angepasst werden. Die Zahl der Trainingsrunden bzw. Epochen muss festgelegt werden. Bei typischen Anwendungen setzt man die Zahl der PSO-Epochen im Bereich 500 bis 2000. Dies stellt die maximale Zahl dar, die der PSO Zeit hat, um das gewünschte Optimum zu finden. In der Regel wird ein Abbruchkriterium definiert, das den Suchprozess terminiert, falls der Algorithmus bereits nach kürzerer Zeit ein Ergebnis gefunden hat, das annehmbar ist oder sich über eine bestimmte Zahl von Epochen keinerlei Verbesserung mehr einstellt. Wählt man die Zahl der Trainingsepochen jedoch zu groß, stellt sich keine Verbesserung mehr ein.

Die Größe jedes einzelnen Schwarms muss ebenfalls vorher festgelegt sein. Zahlreiche Publikationen haben sich mit der optimalen Anzahl der Individuen innerhalb eines Schwarms befasst [6], [5]. Dabei wurde festgestellt, dass die Zahl der Individuen auf etwa 20 bis 25 beschränkt werden sollte. Wählt man den Schwarm größer, erhöht sich der Rechenaufwand enorm, die Ergebnisse werden jedoch nicht schneller oder präziser gefunden. Warum dieses Geschehen eintritt, ist unklar, die Vermutung liegt jedoch nahe, dass bei sehr vielen Individuen oft dieselbe Stelle mehrmals von verschiedenen Partikeln besucht wird.

Die Parameter c_1 und c_2, die soziale und die individuelle Komponente, können ebenfalls variiert werden. Folgt man der Theorie, bewirkt ein Stärken der individuellen Komponente eine Abschwächung der sozialen Komponente, d.h. die Ausrichtung auf die aktuelle global beste Position wird schwächer. Direkt zu beweisen ist diese These aufgrund des stochastischen Charakters des PSO jedoch nur schwer. Überlicherweise werden die beiden Faktoren mit dem Wert zwei vorbelegt, vor allem in [6] finden sich ausführliche Versuche zur optimalen Wahl.

Zusätzlich müssen sinnvolle Grenzen für einen Abbruch des Suchprozesses definiert werden. Beispielsweise besteht die Möglichkeit, dass der Algorithmus bereits nach 20

Epochen das optimale Ergebnis gefunden hat, die maximale Epochenzahl aber mit 500 vordefiniert ist. Ein sinnvolles Abbruchkriterium terminiert den Suchprozess, wenn entweder nach einer bestimmten Epochenzahl keine weitere Verbesserung eintritt oder die Verbesserung nicht mehr groß genug ist. Der zweite Fall tritt oft bei der Optimierung von Funktionen auf: Zwar ist eine Verbesserung immer gegeben, nach langer Suche bewegt sich diese jedoch oft im Nano- oder Picobereich.

Die Dimension des Suchprozesses muss ebenfalls definiert werden. Theoretisch ist die Zahl der Dimensionen dabei unbegrenzt, praktisch haben die meisten Suchprobleme eine Dimension $d \leq 5$, um in möglichst kurzer Zeit zu einem Ergebnis zu kommen. Vor allem bei Echtzeitanwendungen ist dies enorm wichtig.

Zusätzlich zu diesen zwingend notwendigen Parametern gibt es eine Reihe von Fragen und offenen Punkten, die unbestimmt bleiben können, aber in der Regel definiert werden. Hier ist etwa eine "'Geschwindigkeitsbegrenzung"' für die einzelnen Individuen zu nennen. Die meisten Suchräume haben eine feste Größe. Kann die Geschwindigkeit der Partikel nun potentiell sehr groß werden, erschwert das den Suchprozess. Diese Idee ist jedoch umstritten, in der Originalversion von Kennedy/Eberhart existiert diese Begrenzung zunächst nicht, der Algorithmus funktioniert trotzdem. Leichte Verbesserungen sind mit einer Begrenzung der Geschwindigkeit trotzdem möglich, das ist immer von der Anwendung abhängig.

Was mit einem Partikel geschehen soll, das aus dem Suchraum herausdriftet, ist ebenfalls zu definieren. In der Originalversion bleibt dieses Problem ebenfalls offen. Es existieren auch Suchräume, beispielsweise in der Optimierung von Funktionen, die unbegrenzt sind. Ist der Suchraum aber begrenzt, ist diese Definition erforderlich. Mögliche Lösungen sind: Das Individuum prallt am Rand des Suchraumes ab und findet sich auf dem letzten erlaubten Platz wieder, auf dem es zuvor war, das Individuum fällt aus dem Suchraum heraus und tritt von der anderen Seite wieder in diesen hinein oder das Individuum kann den Rand per Definition einfach nicht überschreiten.

2.1.5 Anwendungsmöglichkeiten aus der Praxis

Um dieses theoretische Konstrukt nun mit der Praxis zu verbinden werden im folgenden verschiedene Anwendungsmöglichkeiten und Nutzungsbereiche erläutert.

Optimierung mathematischer Funktionen

Dieser Anwendungsbereich war einer der ersten, für den PSO erfolgreich eingesetzt werden konnte. Es liegt auch nahe, hier PSO zu verwenden, da bei PSO ja stets ein bestimmtes Ergebnis optimiert werden soll. Nun gibt es eine ganze Reihe von Funktionen, deren optimales Ergebnis nur nach langer Zeit gefunden werden kann oder die überhaupt nicht lösbar sind. Gerade solche Funktionen stellen optimale Einsatzgebiete für den PSO dar. Vor allem wenn die Funktion nach einer bestimmten Aktion gleichmäßig gegen einen bestimmtes Ergebnis konvergiert, gelangt der PSO sehr schnell zu sehr guten Ergebnissen. Die gleichmäßige Konvergenz ist für den PSO dabei essentiell, wie später deutlich werden wird.

Am Beispiel einer Funktion soll nun deutlich werden, wie die Optimierung mit PSO funktioniert. Die zu optimierende Funktion Schaffer F6 lautet

$$f(x) = 0.5 + \frac{\sin^2 \sqrt{x^2 + y^2} - 0.5}{(1 + 0.001(x^2 + y^2))^2} \tag{2.9}$$

Diese Gleichung besitzt ein globales Minimum für $x = 0$ und $y = 0$, d.h. $f(x, y) = 0$. Die Verbesserung des Ergebnisses konvergiert dabei systematisch auf diese Werte hin, d.h. wählt man x, y immer wieder kleiner, verbessert sich das Ergebnis in fast jedem Schritt. Diese Funktion stellt in diesem Beispiel also die Fitnessfunktion des PSO dar: In jeder Runde setzt der Algorithmus Werte in die Funktion ein, die ihm von den einzelnen Individuen übermittelt werden. Dabei sorgt die Sozialkomponente in Gleichung 2.1 ständig dafür, dass alle anderen Individuen in Richtung des aktuell gefundenen globalen Optimums driften. Testläufe ergaben, dass der PSO nach spätestens 800 Iterationen einen Wert gefunden hat, der dem Optimum sehr nahe kommt und deswegen einen Abbruch nach sich zieht, z.B. $x, y = 1 \cdot 10^{-6}$. In der Regel sind dann die meisten Individuen in der Nähe des Optimums zu finden, nur einzelne "'Ausreißer"' können sehr weit davon entfernt sein.

Für die Funktion 2.9 muss der PSO zweidimensional sein, jedes Individuum überprüft in jeder Iterationsepoche jeweils einen Wert für x und einen für y. Als weiteres Beispiel sei hier die Griewangk-Funktion genannt, die deutlich zeigt, dass der PSO auch mit komplexen n-dimensionalen Funktionen umzugehen weiß.

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos \frac{x_i}{\sqrt{i}} + 1 \tag{2.10}$$

Neben den beiden genannten Funktionen sind noch Sphere, Rosenbrock oder Rastrigin typische Vertreter gleichmäßig konvergierender Funktionen [6]. Prinzipiell lässt sich natürlich jede erdenkliche Funktion mit dem PSO optimieren. Bei Funktionen, die jedoch keine oder kaum eine Konvergenz bei der Lösungsfindung zeigen, lässt die Performanz des PSO rapide nach. Unter Umständen ist es dann sogar möglich, dass keine oder nur eine sehr weit vom Optimum entfernte Lösung gefunden werden kann. Diese spezielle Eigenschaft des PSO kann jedoch nicht nur zum Nachteil gereichen: Ist es etwa erwünscht, dass eine bestimmte Lösung höchstwahrscheinlich nicht gefunden werden kann, kann dies mit dem PSO realisiert werden. Dass diese Idee für praktische Bildverarbeitungsthemen relevant ist wird im Kapitel 3 ausführlich behandelt.

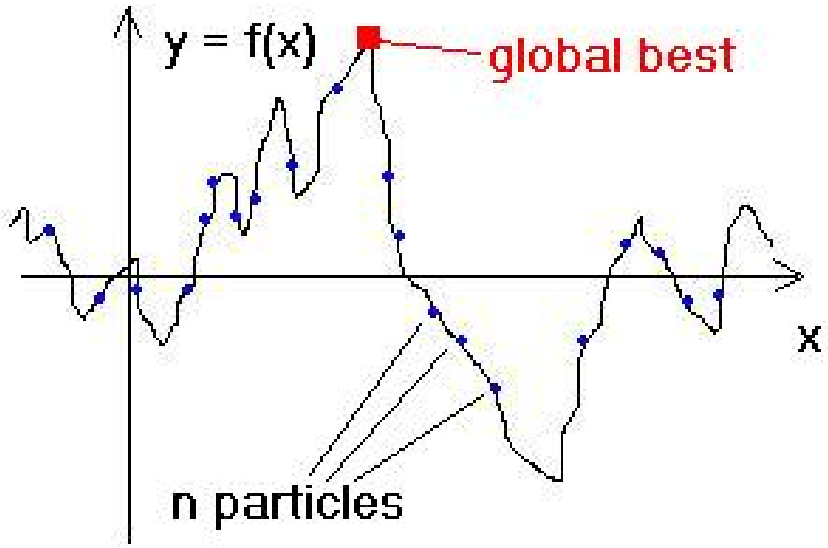

Figure 2.1: PSO Partikel suchen nach der global besten Position der Funktion

Object Tracking

Unter dem englischen Sammelbegriff "'Tracking"' verbirgt sich das Nachführen oder Verfolgen sich bewegender Objekte in einer Videosequenz oder einer Bilderfolge. Dieser allgemein bekannte und meistverwendete Begriff wird im Folgenden verwendet.

Das Tracking von Objekten oder Personen in Videos läuft dabei folgendermaßen ab: Zunächst muss irgendein Kriterium existieren, das ein bewegtes Objekt vom Hintergrund unterscheidet. Oft werden dazu zwei aufeinanderfolgende Frames voneinander

Figure 2.2: Exemplarische Tracking-Szene mit zwei Personen

abgezogen, an den statischen Stellen im Bild ergeben sich dann sehr kleine Werte, an den übrigen relativ große. Dieses relativ einfache Verfahren funktioniert nur bei statischem Hintergrund.

Das eigentliche Suchkriterium für den PSO muss nun wieder über eine zu definierende Fitnessfunktion vorgegeben werden. Eine sehr einfache Fitnessfunktion kann beispielsweise durch simple Abfrage des Grauwertes an der betroffenen Stelle erstellt werden. Bei einem digitalen Bild ist schließlich jedem Pixel ein fester Grauwert zugeordnet. Startet man nun den PSO, wird das zweidimensionale Suchfeld durchlaufen. Hier sind natürlich verschiedenste Fitnessfunktionen denkbar, etwa könnte man zusätzliche extrahierbare Kriterien hinzunehmen oder eine größere Suchmaske (z.B. 9 Pixel groß) wählen.

Weitere Anwendungen

Eine weitere Anwendungsmöglichkeit für PSO stellt das Training für neuronale Netze dar [9]. Ein künstliches neuronales Netz (KNN) versucht, grundlegende Prozesse im

menschlichen Gehirn nachzubilden. Dieses KNN besteht aus mehreren Neuronen, die untereinander vernetzt sind. Bei Eintritt bestimmter Ereignisse kann jedes Neuron aktiviert werden, diesen Prozess bezeichnet man als "'Feuern"'. Um dieses KNN funktionstüchtig zu machen ist eine umfangreiche Trainingsphase notwendig. In dieser Phase muss dem KNN automatisiert oder per Hand anhand von Beispielen vorexerziert werden, was als "'richtig"' und "'falsch"' oder "'passend"' und "'unpassend"' anzusehen ist. Dieser Prozess kann durch PSO automatisiert werden: In der Fitnessfunktion kann definiert werden, was als richtig oder falsch angesehen wird. Nun kann der PSO systematisch nach brauchbaren Lösungen suchen und diese dem KNN mitteilen. Genauso können "'Nieten"', also Ergebnisse, die unerwünscht sind, an das KNN als unerwünscht weitergeleitet werden.

Eine weitere praktische Anwendung ist das sogenannte "'Finite Element Updating"', bei dem nach einem Ereignis von einem Initialmodell auf ein anderes Modell geschlossen werden kann. Beispielsweise verfügt jeder Autohersteller über detaillierte Computermodelle seiner Fahrzeuge. Nun will er wissen, wie sich dieses Modell bei einem bestimmten Aufprall mit einer bestimmten Geschwindigkeit verhält. Nach der Einspeisung seines Modells kann er dafür Finite Element Updating verwenden.

Im Folgenden soll auf den Forschungsbereich Tracking und was dort bereits realisiert ist, eingegangen werden. Dieser praktischen Anwendung von PSO ist diese Forschungsarbeit gewidmet.

2.2 State of the Art im Bereich Tracking

Spätestens seit den achtziger Jahren, in denen in Deutschland neben ersten leistungsfähigen Computern im Heimanwenderbereich auch Videogeräte erschwinglich wurden, existiert die Idee, bestimmte Merkmale aus bewegten Bildern zu extrahieren und zu verfolgen. Die zahlreichen Anwendungsmöglichkeiten kristallisierten sich in den folgenden Jahren immer stärker heraus: Etwa für Sicherheitssysteme im militärischen Bereich, in dem dann Personal eingespart werden kann, in Unterhaltungssystemen wie etwa der neuen Spielkonsole Nintendo Wii, bei der Interaktion durch Bewegungen mit dem Controller oder durch manuelles Nachführen bestimmter Objekte entsteht oder seit einiger Zeit auch in Notfallsystemen für alte oder behinderte Leute, bei denen die

Software selbstständig Kamerabilder auswerten und unnatürliche Haltungen des Körpers als Notfall erkennen muss, beispielsweise einen Sturz auf den Boden.

2.2.1 Kantendetektion

Um überhaupt etwas aus einem Bild extrahieren zu können, war es zunächst nötig, bestimmte Filter oder Masken zu entwickeln, die in einem digitalen Bild verschiedene Merkmale hervorheben bzw. extrahieren können. Exemplarisch sei hier die Kantendetektion genannt: Durch Faltung einer bestimmten Maske mit den Grauwerten des digitalen Originalbildes entsteht eine starke Betonung der Kanten, idealerweise sind nur noch die Kanten sichtbar. So war es erstmals möglich, starke Übergänge in einem Bild, wie sie etwa zwischen einer Person und ihrem Hintergrund auftreten, zu extrahieren. Entwickelt wurden solche Kantenfilter von Sobel oder Canny [10], die zu den meistverwendeten Algorithmen in diesem Bereich zählen. Von dieser Idee existieren zahlreiche Varianten, etwa könnte ein Sobel-Operator zur Kantenfilterung in x-Richtung folgendermaßen aussehen:

$$G_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \star A \tag{2.11}$$

Wird diese zweidimensionale Maske nun mit einem digitalen Bild A gefaltet, entstehen an Kanten und starken Übergängen unnatürliche Verstärkungen, an allen übrigen Bereichen wird das Bild dagegen relativ gleichmäßig schwarz. Wählt man die Maske größer, verhält sie sich unempfindlicher gegenüber Störungen wie beispielsweise Rauschen. Einzelne Peaks, die durch Rauschen entstehen, führen dann durch den weiter gefassten Sobel-Filter seltener zu Kanten.

Abbildung 2.3 zeigt ein Beispiel für den grundsätzlichen Ablauf einer Extraktion zweier sich bewegender Objekte, im Beispiel von zwei Personen. Unten rechts ist eines der original Grauwertbilder zu sehen. Zuerst werden nun die zwei Frames vom RGB-Farbraum in Grauwerte umgewandelt, dann voneinander subtrahiert. Die Umrisse der ersten Person sind bereits zu erkennen, das Bild ist jedoch von starkem Rauschen überlagert. Das Bild links oben zeigt den entstandenen Frame nach Filterung mit einem Medianfilter (Werte hier [3 3]), der das Rauschen gut unterdrückt. Allerdings wird durch das Glätten mit einem Medianfilter auch eine größere Unschärfe der Kanten in Kauf

Figure 2.3: Anwendungsbeispiel eines Sobel-Operators

genommen. Zur Kantendetektion wird anschließend ein Sobel-Filter verwendet; es werden jeweils die Stellen detektiert, an denen der Bildgradient ein Maximum aufweist (Bild rechts oben). Der Kopf der Marktfrau als auch die meisten Körperteile der anderen Person sind sichtbar.

Nun wird noch eine zweidimensionale Faltung durchgeführt, um die Kanten noch deutlicher zu machen (hier wurde das Bild nach der Sobel-Filterung mit der Maske aus Gleichung 2.12 gefaltet). Nun könnte man beispielsweise mit einem Schwellwertfilter alle hellen Pixel extrahieren und so beide Personen verfolgen.

$$M_f = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.12}$$

Es ist mithilfe der Kantendetektion also möglich, einfache Bildverarbeitung durchzuführen. Allerdings werden solche Verfahren in heutiger Software weniger als

alleinige Methode verwendet, sondern eher als Vorverarbeitung für die eigentliche Merkmalsextraktion. Dies liegt vor allem darin begründet, dass ein statischer Kantenfilter alleine heutigen Anforderungen in Bezug auf Helligkeitsänderungen, dynamischen Hintergründen oder sehr schnellen Bewegungen nicht mehr gerecht wird. Durch starke Helligkeitsänderungen, etwa das Ein- und Ausschalten der Beleuchtung im Versuchsraum, werden die Kanten der Objekte unter Umständen stark verwischt. Ähnlich verhält es sich bei schnellen Bewegungen der Objekte, besonders wenn das Kamerabild im Halbzeilenverfahren (Interlaced Verfahren) aufgenommen wurde. Durch das Zusammensetzen der beiden Halbbilder entstehen oft unscharfe Objekte, deren Kanten dann oft fließend in den Hintergrund übergehen.

2.2.2 Blockmatching

Dem Blockmatching liegt ein Bewegungsmodell zugrunde, das Helligkeitsänderungen von Pixeln als Bewegung interpretiert und diese durch eine skalare Verschiebung von Pixeln in x- und y-Richtung darstellt [11]. Dabei können mit Blockmatching Operationen wie Rotationen, Größenänderungen oder Zooms nur eingeschränkt beschrieben werden.

Zunächst muss das gesamte digitale Bild in Blöcke einer bestimmten Größe unterteilt werden, etwa 8x8 oder 8x4 Pixel sind typische Blockgrößen (bei H.264/MPEG-4 AVC). Umso größer die Blöcke, umso weniger Rechenaufwand ist nötig, die Suche wird jedoch auch ungenauer, etwa können Bewegungen sehr kleiner Objekte (z.B. ein Finger einer Person) unter Umständen gar nicht mehr verfolgt werden. Bewegungen sind mit größeren Blöcken generell ungenauer modellierbar, hier muss die Blockgröße möglichst an die Videosequenz angepasst werden. Kleinere Blöcke enthalten mehr Bewegungsvektoren, haben einen kleineren Restfehler, modellieren die Bewegungen genauer und haben pro Block weniger Bewegung zu verarbeiten [11]. Bei neueren Codierungen, etwa H.264/MPEG-4 AVC ist auch eine Auflösung auf Subpixel-Ebene möglich, d.h. die Genauigkeit ist nicht auf die Pixelabstände beschränkt.

Die Funktionsweise der Algorithmen beruht nun auf diesen Blöcken: Banal ausgedrückt "'merkt"' sich der Algorithmus einen bestimmten Block aus Frame t und sucht diesen Block im Frame $t + 1$ oder einen möglichst ähnlichen wieder. Dies wird mit allen Blöcken durchgeführt, die Verschiebung jedes Blocks wird mit einem Vektorpfeil sichtbar im Bild markiert. Die maximale Länge eines Bewegungsvektors wird dabei

durch den Suchraum bestimmt: müsste der Algorithmus jedesmal das gesamte Bild für jeden Block durchsuchen, wären diese Algorithmen viel zu langsam. Deshalb wird der Suchraum auf einen bestimmten Bereich, der meistens in der Nähe des letzten Aufenthaltsortes dieses Blockes liegt, begrenzt.

Bleibt noch zu klären, welche Kriterien Blockmatching Algorithmen zur Detektion der Blöcke verwenden. In der Regel werden dafür ausschließlich Helligkeitskriterien verwendet, also z.B. Grauwerte. Ein digitales Grauwertbild enthält an jeder Pixelkoordinate einen eindeutig zugewiesenen Wert, dieser liegt, je nach Farbtiefe, z.B. zwischen 0 und 255. Mit den bereits angesprochenen Verfahren MSE (Mean Squared Error) oder SAD (Sum of Absolute Differences) kann anschließend die Ähnlichkeit zweier Blöcke bestimmt werden.

$$MSE(v,h) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}[I_{t-1}(i+h, j+v) - I_t(i,j)]^2 \tag{2.13}$$

$$SAD(v,h) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|I_{t-1}(i+h, j+v) - I_t(i,j)| \tag{2.14}$$

Die Formeln 2.13 und 2.14 stellen die zwei Verfahren dar. I_{t-1} ist die Intensität zum Zeitpunkt $t-1$, I_t zum Zeitpunkt t. Die Parameter v und h stellen die Verschiebung des aktuell zu prüfenden Elementes dar, i, j sind die urspünglichen Positionen des Blockes. Die Ergebnisse werden Pixelstellenweise errechnet und danach innerhalb eines Blockes aufsummiert.

Weiterhin gibt es im Bereich Blockmatching noch Verfahren, um Blöcke innerhalb des Suchbereiches nicht zufällig durchprobieren zu müssen, sondern passende gezielt herauszufiltern. Hier gibt es sehr viele Möglichkeiten, eine bekannte ist der Three Step Search(TSS): dieses Verfahren vergleicht die horizontalen, vertikalen und diagonalen Punkte im Abstand der Startschrittweite, vergleicht jeweils den Fehler, geht in Richtung des geringsten Fehlers und verringert die Schrittweite. Dann wird erneut gesucht [11].

Blockmatching stellt ein relativ einfach realisierbares Verfahren dar, das jedoch über zahlreiche Schwächen verfügt: Außer den Helligkeitsstufen sind keine Kriterien möglich, bei einer starken Änderung der Lichtverhältnisse ist das Verfahren somit sehr schwach, die Blockgrößen sind in der Regel fest und können damit kaum auf unterschiedlich schnelle Bewegungen reagieren, der Suchraum ist meistens ebenfalls von fester Größe.

2.2.3 Hidden Markov Modelle (HMM)

Hidden Markov Modelle sind Netze, deren Knoten Zustände und deren Kanten Übergänge darstellen [12]. Auch hier handelt es sich wie bei PSO um ein stochastisches Modell, d.h. die Aktionen eines HMM sind letztendlich nicht vorhersehbar. Ein untrainiertes HMM kann man sich wie einen neugeborenen Menschen vorstellen: Bis auf einige Grundfertigkeiten, die aus der Struktur bedingt sind, ist dieser zunächst relativ handlungsunfähig. Die geistigen Fähigkeiten entstehen erst durch die Ausbildung von Neuronen im Gehirn und durch jahrelanges Training. Analog dazu bedingt ein HMM ebenfalls einen enormen Trainingsaufwand.

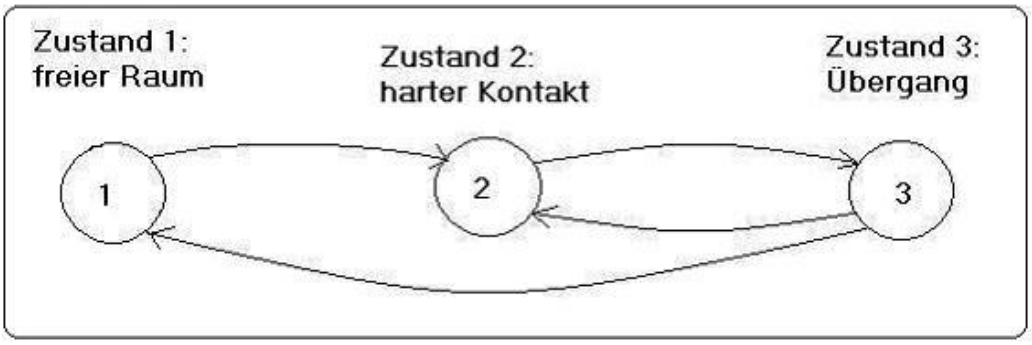

Figure 2.4: Exemplarische Markovkette

Abbildung 2.4 soll die Grundstruktur eines Markov-Modells verständlich machen. Die drei Kreise stellen die Zustände dar, die Pfeile sind jeweils die Übergänge. Jeder Zustand des Modells besitzt eine Emissionswahrscheinlichkeit b_i [12]. Tritt die Emission eines Zustandes nun ein, ist dies gleichbedeutend mit dem Eintritt des Ereignisses, das mit dem Zustand per Definition verbunden ist. Im gewählten Beispiel in Abbildung 2.4 handelt es sich um einen Roboter, der autonom in einem Raum navigiert. Dieser Roboter könnte mit einem Drucksensor, beispielsweise einem Piezoelement, einen Zusammenstoß mit einem Objekt feststellen. Zustand 1 bedeutet nun, dass kein Kontakt mit irgendeinem Hindernis vorliegt, Zustand 2 stellt einen Kontakt dar, Zustand 3 ist ein Übergangszustand, der z.B. nach einem Zusammenstoß eintritt. Das HMM ist nun in der Lage, diese Vorgänge zu "'schätzen"', wenn es Daten zur Verfügung hat und lange genug trainiert wurde. Exemplarisch könnte der Roboter einen Sensor haben, der den Rollwiderstand des Roboters misst. Nehmen wir weiterhin an, dass der Teppichboden in der Nähe der Raumecken welliger ist als in der Mitte des Raumes, der

Roboter würde dann in den Ecken schwerer vorankommen. Würde man diesen Vorgang nun lange genug mit einem HMM trainieren, würde das Modell irgendwann den Zusammenhang welliger Boden ist wahrscheinlich Zustand 2, glatter dagegen Zustand 1, erkennen. Ein Kritiker mag nun einwenden, dass in einem solchen Fall ein simpler Drucksensor, etwa ein Piezoelement, schneller und sicherer zum gewünschten Ergebnis führt. Das mag im gewählten Beispiel richtig sein, dennoch gibt es sehr viele Probleme, die sich nicht oder nur mit enormem Aufwand direkt und deterministisch lösbar sind und für die Hidden-Markov-Modelle unerlässlich sind.

Etwa im Bereich Tracking ist das HMM ein oft verwendeter Kern des Algorithmus. Ein HMM kann beispielsweise sagen, dass es sich bei einem gefundenen Objekt mit einer Wahrscheinlichkeit von 85 Prozent um eine Person handelt. Dem Programmierer bleibt es dann überlassen, ab einem bestimmten Wert einen Treffer anzuzeigen. Oft ist es bei solchen Aufgabenstellungen auch nötig, das HMM in mehrere kleinere Teilereignisse zu zerlegen und anschließend eine Gesamtwahrscheinlichkeit zu bilden. So arbeitet das HMM dann wesentlich genauer. Beispielsweise hat die Person, die zu detektieren ist, ein rotes Hemd und eine blaue Hose an. Diese Aufgaben können getrennt werden, d.h. es existiert dann eine Wahrscheinlichkeit für das Detektieren des roten Hemds und eine für das Detektieren der blauen Hose. Aus diesen Einzelwahrscheinlichkeiten muss nun eine Gesamtwahrscheinlichkeit gebildet werden, etwa mit dem Baum-Welch-Algorithmus oder dem Viterbi-Algorithmus [12]. Etwa in [13] wird aktuell das Konzept des HMM für komplexe Tracking-Anwendungen benutzt.

Hidden-Markov-Modelle bieten also ein Reihe von Vorteilen: Es wird stets ein stochastisches Ergebnis ausgegeben, nach längerem Training sind sehr gute Ergebnisse möglich [13]. Nachteile sind der hohe Trainingsaufwand, die Unflexibilität bei starken Lichtschwankungen und der oft sehr hohe Aufwand, ein detalliertes HM-Modell zu entwerfen.

2.2.4 Optischer Fluss

Als optischen Fluss bezeichnet man in der Bildverarbeitung und der optischen Messtechnik ein Vektorfeld, das die Bewegungsrichtung für jedes Pixel innerhalb des Bildes angibt. Hier existieren wieder zahlreiche Abweichungen, etwa kann statt der Bewegung jedes Pixels auch nur die Bewegung von Pixelblöcken bestimmter Größe

verfolgt werden, etwa beim Moving Picture Experts Group (MPEG) Standard ist dies der Fall.

Zunächst ist der Optische Fluss wieder ein Verfahren, dessen Ergebnisse statt direkt berechnet zu werden geschätzt werden müssen. Bei den Schätzungen werden als einziges verschiedene Helligkeitsmuster im Bild verwendet. In der Regel benutzt man auch hier die Grauwerte des digitalen Bildes, um Helligkeitsstufen zu bekommen. Anschließend sucht dieses Verfahren starke Helligkeitsgradienten in der Nähe der Stelle, an der sich ein bestimmtes Helligkeitsmuster zuletzt befand. Ein Gradient ist dabei ein Vektor, der in einem Skalarfeld immer in Richtung des stärksten Anstiegs weist. Legt man dabei einen festen Startpunkt fest, existiert nur ein Gradient, der in Richtung des stärksten Anstiegs der Grauwerte zeigt. Muss der Startpunkt jedoch nicht fest sein, kann man mehrere Gradienten erschließen.

Der grundsätzliche Lösungsansatz dieses Prinzips geht auf Berthold Horn und Brian Schunck zurück [14]. Man nimmt an, dass die Helligkeit E an entsprechenden Stellen der Einzelbilder in der Bildsequenz konstant ist. Anschließend wird die Bestimmungsgleichung für die Geschwindigkeiten gelöst: Es entsteht ein lineares Gleichungssystem, das z.B. mit der Jacobi Iteration gelöst werden kann [15].

Der Optische Fluss ist also ein Verfahren, das die Bewegungen im Bild mit Vektoren sichtbar machen will. Einige Standards wie z.B. MPEG benutzen Teile dieses Verfahrens, um Komplexität einzusparen. Ein Einsatz im Tracking-Bereich ist ebenfalls möglich, allerdings sind die nötigen Rechenprozesse durch die vielen Gradienten relativ komplex. Dennoch liegt das Hauptproblem nicht in den Rechenzeitanforderungen, sondern in der Erkennungsleistung. Bei Helligkeitsschwankungen oder abrupten, schnellen Bewegungen lässt die Performanz des Optischen Flusses rapide nach.

2.2.5 Weitere Methoden

Neben den genannten existieren noch zahlreiche weitere Methoden, die den Stand der Technik im Bereich Objektnachführung darstellen.

Hier ist etwa Mean Shift zu nennen, ein Verfahren, das die Verteilung der Helligkeitsgradienten schätzt und daraus Bewegungsrichtungen ermittelt, etwa in [16]. Ein weiteres Verfahren stellt der Kalman Filter dar, ein stochastischer Zustandsschätzer für dynamische Systeme. Der Kalman Filter verwendet dabei systematisch die Ergebnisse der letzten Runden, um aktuelle Ergebnisse zu schätzen. Er trainiert sich also während

des Arbeitsprozesses und wird in der Regel immer besser, umso mehr Daten aus der Vergangenheit vorliegen. Eine Einführung in die Theorie des Kalman Filters wird z.B. in [17] gegeben.

Chapter 3

Realisierung

3.1 Realisierung einer PSO-basierenden Tracking Software

Für die umfassende Forschungsarbeit war es zunächst nötig, eine grundständige PSO-basierende Tracking Software zu programmieren. Dies geschah mit der Programmiersprache MatLab, Version 7.0.

3.1.1 Aufnahme von Videosequenzen

Um verschiedenste Situationen und Möglichkeiten testen zu können, war es unumgänglich, eine größere Anzahl an Videosequenzen selbst aufzunehmen. Für den Anfang gibt es zwar durchaus frei verfügbare, brauchbare Sequenzen, um verschiedene Extremsituationen darzustellen muss jedoch selbst aufgenommen werden. Zu diesen Situationen zählt etwa das kurzzeitige Verschwinden des Objektes aus dem Bild, starke Helligkeitsschwankungen (können z.B. durch Ein- und Ausschalten der Raumbeleuchtung herbeigeführt werden) oder schnelle, abrupte Bewegungen. Solche unerwarteten Bewegungen werden von der Fachliteratur als "'rigid"' oder "'non rigid"' bezeichnet [8], allerdings ist es schwer, zwischen diesen undeutlichen Definitionen eine Grenze zu ziehen.

Die Videosequenzen decken also eine ganze Reihe verschiedener Situationen ab, die anschließend neben Standardsequenzen zum Tracking eingesetzt werden sollen. Der Versuchsaufbau gestaltete sich wie folgt: für die Aufnahmen wurde eine MD-9035 Digitalkamera verwendet, sie liefert 25 fps bei einer Auflösung von 720x576 Pixel (PAL).

Hier wurden verschiedenste Szenarien ausprobiert, etwa mit Personen im Bild, Objekten (Bällen, Figuren etc.) oder anderweitigen Gegenständen. Stark schwankende Lichtverhältnisse wurden ebenso simuliert wie sehr schnelle und unvorhersehbare Bewegungen, etwa das Fallenlassen eines Balles.

3.1.2 Einlesen der Sequenzen

Für die Verwendbarkeit der Videosequenzen in Matlab war es nötig, diese einzulesen. Anschließend stellte sich die Frage, ob man jeden oder nur jeden n-ten Frame für eine Trackingsequenz benötigt. Diese Problemstellung ist nicht einfach zu beantworten und immer abhängig von der Sequenz, bei schnellen Bewegungen ist z.B. eine höhere Framerate nötig als bei langsamen. Dieses Thema wird später im Paragraph adaptive Fitnessfunktion noch genauer erläutert, die Grundidee ist folgende: Falls es möglich wäre, die Geschwindigkeit, mit der sich Objekte bewegen, abzuschätzen, könnten adaptiv mehr oder weniger Frames pro Sekunde verwendet werden.

Zunächst muss also eine abgeschlossene Sequenz in Matlab eingelesen werden, die dann analysiert werden kann. Zwar ist es mit Matlab auch möglich, Videosequenzen in Echtzeit zu bearbeiten, dies ist jedoch nur mit speziellen Kameras und Verbindungen, die schnell genug arbeiten, möglich. Beim Einlesevorgang gilt es nun, einen möglichst guten Kameracodec zu verwenden. Matlab kann standardmäßig keinen DV-Codec, der üblicherweise bei Kameras verwendet wird, einlesen. Deshalb ist man auf einen Codec der diversen Kamerahersteller, z.B. Phillips, angewiesen, wenn man diese Videos einlesen will. Hier sollte möglichst ein Codec verwendet werden, der gute Reaktionen bei Bildern im Halbzeilenformat aufweist, d.h. der Interlaced-Videomaterial möglichst "'nahtfrei"' wieder zusammensetzt, auch bei schnellen Bewegungen. Weiterhin sollte der Einleseprozess möglichst schnell vonstatten gehen um einen späteren Einsatz in Echtzeitanwendungen zu ermöglichen.

3.1.3 Codierung eines State of the Art PSO

Grundstruktur

Nach dem Einlesen müssen die RGB-Frames in Grauwertstufen umgewandelt werden. Anschließend werden jeweils zwei aufeinanderfolgende Grauwertframes voneinander subtrahiert, so können in einem Video mit weitgehend statischem Hintergrund die Be-

wegungen der Objekte extrahiert werden. Die nun entstandenen Bilder werden noch mit einem Medianfilter gefiltert um normalverteiltes Rauschen aus dem Material zu bekommen. Um dem PSO nun Startpunkte für die Individuen vorgeben zu können, müssen die ersten beiden Frames konservativ durchsucht werden. In der Literatur gibt es zwar zahlreiche Ansätze, die den Startpunkt der Individuen zufällig vorgeben [5], dies führt jedoch für diese Anwendung nicht zum Erfolg. Diese Entdeckung war für die spätere Weiterentwicklung zum Self-Splitting PSO wegweisend. Bei einer Auflösung von 720x576 Pixel existieren 414720 Punkte im gesamten Bild. Lässt man nun von jedem Individuum ein Pixel pro Epoche überprüfen kommt man auf 20 Pixel pro Epoche, überprüft jedes Pixel einen Block, der aus z.B. 4 Pixeln besteht, kommt man auf 80 Pixel pro Epoche. Diese Verhältnisse zeigen deutlich, dass der PSO mit zufälligen Startpunkten in diesen Größenverhältnissen nicht viel ausrichten kann. Dazu kommt noch, dass ein Großteil des Bildes nach dem Abziehen zweier aufeinanderfolgender Frames den Wert 0 hat (ideales, rauschfreies Bild vorausgesetzt), nämlich an allen Stellen, an denen der statische Hintergrund gleichgeblieben ist. Setzt man nun ein PSO Individuum in einen Suchraum, der in der näheren Umgebung fast ausschließlich aus Nullen besteht, kann das Individuum nichts finden. Diese Tatsache wurde durch zahlreiche Versuche verifiziert, das lebenswichtige konvergierende Verhalten fehlt dem PSO. Zwar handelt es sich bei PSO um ein sehr mächtiges Suchverhalten, das jedoch völlig nutzlos sein kann, wenn konvergierendes Verhalten fehlt.

Im Laufe der Versuche wird sich jedoch zeigen, dass dieses Verhalten Vorteile bringen kann, wenn sich mehrere Objekte im Suchraum befinden, die weit genug voneinander entfernt sind. Beispielsweise soll Objekt 1 nachgeführt werden, Objekt 2 ist ebenfalls im Bild. Objekt 2 ist durch die Fitnessfunktion als "'besseres"' Ziel definiert. Dennoch kann es nie gefunden werden, wenn der Abstand zwischen den Individuen und Objekt 2 zu groß ist. Es ist also möglich, die Individuen sehr gezielt in ein lokales Maximum laufen zu lassen wenn das globale Maximum zu weit entfernt ist. Dieser Zusammenhang wird später noch ausführlicher diskutiert.

Die Startpunkte für die Individuen müssen also vorgegeben werden, nur so kann sichergestellt werden, dass sich die Individuen in der nächste Runde, also bei den nächsten Frames, sehr nahe am gesuchten Objekt befinden. Innerhalb des gesuchten Objektes ist dann in der Regel das geforderte, konvergierende Verhalten zu finden, dies hängt

Parameter	Wert	Bedeutung
Me	2000	maximale Anzahl der Trainingsepochen
Ps	20	Anzahl der Individuen pro Schwarm
ac1	2	individuelle Komponente
ac2	2	soziale Komponente
iw1	0.9	initialer Gewichtungsfaktor
iw2	0.4	finaler Gewichtungsfaktor
iwe	500	Epochenzahl, nach denen iw2 erreicht wird
ergrd	0.9	Abbruchbedingung bzw. minimal nötige Verbesserung
D	2	Zahl der Dimensionen des Suchraums
mv	16	maximale Geschwindigkeit eines Individuums

Table 3.1: Parameter des PSO

damit zusammen, dass das Objekt an den Rändern meistens schwächere Änderungen besitzt als in der Mitte, dieser Punkt wird später noch genauer erläutert.

Zurück zur Implementierung. In der ersten Runde wird das Objekt also herkömmlich nachgeführt, d.h. durch Extraktion von Grauwerten, die sich besonders stark verändert haben. An diesen Stellen muss sich das sich bewegende Objekt befinden. In der nächsten Runde werden nun diese "'Volltreffer"' der letzten Runde als Startpunkt verwendet. Selbst wenn sich das Objekt mit sehr hoher Geschwindigkeit bewegt, ist es in $\frac{1}{25}$ Sekunde nicht möglich sich sehr weit von den genannten Punkten zu entfernen. Die Individuen finden in der nächsten Suchrunde also ganz in der Nähe (was "'ganz in der Nähe"' bedeutet, ist von vielen Faktoren abhängig wie Auflösung des Bildes, Suchgebiet und Geschwindigkeit der Individuen etc.) einen Bereich mit großen Zahlenwerten, der zudem das geforderte konvergente Verhalten aufweist. Dies stellt ideale Bedingungen für den PSO dar, das Objekt wird problemlos nachgeführt.

Definieren der Parameter

Die bereits besprochenen, grundlegenden Parameter des PSO müssen definiert werden, einige typische Werte kann man aus [6] oder der Toolbox von Brian Birge entnehmen [18]. Folgende Parameter mit ihren Werten wurden dabei definiert:

Vor allem die Dimensionierung der maximalen Partikelgeschwindigkeit sollte genau vorgenommen werden: Dieser Parameter kann sehr nützlich sein, wenn man beispielsweise den Suchraum begrenzen oder die Partikel in einen bestimmten Bereich "'lotsen"' will. Die Dimension ist bei Tracking Anwendungen in der Regel zweidimensional, eventuell dreidimensional wenn man Farbbilder verwendet und z.B. den Hue-Wert im HSV-Farbraum als dritte Dimension auffassen will. Die Größe des Schwarms ist variabel, jedoch stellen sich bei Werten über 30 keine oder kaum mehr Verbesserungen ein [6].

Testergebnisse

Es ist wie bereits gesagt nötig, den ersten Schritt der Trackingsequenz nicht durch PSO, sondern durch ein herkömmliches Verfahren zu tracken. Die Punkte mit hoher Helligkeitsänderung zwischen zwei Frames werden im ersten Schritt gesucht und diejenigen, die höher als ein bestimmter Schwellwert liegen, ausgewählt. Die Kunst dabei besteht darin, den Schwellwert richtig festzulegen. Setzt man den Schwellwert etwa zu niedrig an, erhält der PSO Algorithmus keine optimalen Startwerte, u.U. findet er dann nicht das absolute Maximum, sondern nur ein lokales.

Die gefundenen Punkte werden dem PSO dann als Startwert für die Partikel des Schwarms übergeben. Nun kann es natürlich vorkommen, dass die Vorverarbeitung z.B. 40 Koordinatenpaare liefert, der Schwarm aber nur aus 20 Partikeln besteht. Hier werden dann einfach die ersten 20 verwendet, man erhält keine besseren Ergebnisse, wenn man die 20 hellsten unter den z.B. 40 auswählt. Liegt die Anzahl der Werte der Vorverarbeitung erheblich über der Größe des Partikelschwarms, z.B. bei 500, muss der Threshold-Wert angepasst werden. Mit diesem Wert definiert man die Grenze zwischen "'brauchbaren"' und "'nicht brauchbaren"' Helligkeitsstufen.

Die Partikel des PSO starten nun also in der Nähe der Person, da in einem Sekundenbruchteil (z.B. bei 5 oder 10 Frames pro Sekunde, die getrackt werden) nur eine geringe Bewegung möglich ist. Von dieser Position aus haben die Partikel dann optimale Chancen, die Bewegungen der Person nachzuführen. Der PSO startet dann also mit vorgegebener Initialposition, aber zufälliger, variabler Anfangsgeschwindigkeit (diese passt sich dann nach wenigen Trainingsrunden automatisch an die Aufgabe an). Eine vorgegebene maximale Geschwindigkeit kann dabei sinnvoll sein, hier wurde $V_{max} = 16$ verwendet.

Runde	x	y	Grauwertdifferenz
1	265	35	0
2	265	35	0
3	261	35	4
4	251	35	81
5	250	36	89
6	251	37	89

Table 3.2: Annäherung des PSO an das global beste Ergebnis

Die Ergebnisse eines Testdurchlaufes (5 Frames pro Sekunde, Video hat 25 fps, es wird also jeder fünfte Frame zur Analyse herausgegriffen, Auflösung 288x213) sehen etwa folgendermaßen aus: Dauer eines Durchlaufes für 2 Sekunden Video (also Analyse und Tracking von 10 Frames) 1,05 Sekunden, also echtzeitfähig (auf Intel Pentium 1,6 GHZ, 496 MB Ram). Exemplarisch seien hier die ersten Runden des Annäherungsprozesses dargestellt:

Nach der ersten Trainingsrunde ist die globale beste Position das Koordinatenpaar 265/35 mit der Helligkeitsänderung 0. Nach erfolgtem dritten Schritt ist das erste $gbest > 0$ gefunden: Es liegt bei dem Koordinatenpaar 261/35. Dieses $gbest$ weist einen geringeren y-Wert auf als alle vorherigen gbests (261 statt 265). Die anderen Partikel "'merken"' diesen verbesserten Wert und driften bei der Geschwindigkeitswahl für die neue Trainingsrunde verstärkt in die negative y-Richtung bzw. in Richtung dieser Koordinate, falls sie weit weg vom global best sind. Das Suchen in dieser Richtung bringt in der nächsten Runde schon fast einen Volltreffer: Der Helligkeitsunterschied zwischen den 2 Frames liegt hier bei 81, ein sicheres Zeichen, dass die Person an dieser Stelle im Bild ist. Mit dem Wert 89 wird anschließend das global best des Bildes gefunden. Wird nun 250 Trainingsrunden lang kein höherer Wert mehr gefunden, terminiert der Algorithmus. Insgesamt sind maximal 2000 Trainingsrunden vorgesehen, um das global best zu finden.

Ein Anheben der maximalen Geschwindigkeit brachte bessere Ergebnisse: Jedes Partikel kann seine Geschwindigkeit in jeder Dimension frei wählen, es wird nur eine bestimmte Maximalgeschwindigkeit vorgegeben, die ein völlig unkontrolliertes Auseinanderdriften des Schwarmes verhindern soll: Die obigen relativ guten Ergebnisse wurden mit der maximalen Geschwindigkeit 16 erzielt, d.h. jedes Partikel kann in jeder Rich-

tung pro Runde maximal 16 Pixel zurücklegen. Diese Geschwindigkeit ist natürlich abhängig von z.B. der Bildschirmauflösung und somit niemals optimal, sondern immer nur näherungsweise sehr gut.

Testdurchläufe mit verschiedenen Fitnessfunktionen

Zunächst wurde eine sehr einfache Fitnessfunktion verwendet: Das Individuum untersucht nur die Pixelposition, an der es sich gerade befindet, d.h. die Helligkeit eines einzelnen Pixels entscheidet in diesem Fall über die Wertigkeit einer bestimmten Position. Hier wurden auch Untersuchungen mit komplexeren Fitnessfunktionen durchgeführt. Eine Fitnessfunktion kann nur aus einer relativ kurzen Angabe, aber auch aus einer komplizierten Funktion bestehen. So ist es auch möglich, verschiedene Informationen unterschiedlich stark zu gewichten, z.B. kann der Grauwert zu 70 Prozent und der Rotanteil im RGB-Farbraum zu 30 Prozent das Ergebnis bestimmen. Auch ist es möglich, die Funktion aus mehreren Informationen zusammenzusetzen, z.B. das Ergebnis von vier oder neun Pixeln. Das Gesamtergebnis aus diesem Block wird dann aus der Addition der Einzelergebnisse gewonnen. Diese Idee, zwei Frames voneinander abzuziehen und danach einen Block bestimmter Größe zu addieren entspricht fast genau den Ansätzen MSE und SAD, die bereits im Paragraph Blockmatching ausführlich behandelt worden sind.

Umfangreiche Tests ergaben ein differenziertes Bild: Während bei einfacheren Tracking Aufgaben eine Änderung der Fitnessfunktion oft nicht sinnvoll ist (eine sehr einfache Werteabfrage reicht dann in der Regel aus), kann es bei schwierigen Aufgaben mit abrupten Bewegungen und diffusen Lichtverhältnissen sinnvoll sein, die Abfragen an die gegebenen Bedingungen anzupassen. Ebenfalls sinnvoll ist eine adaptive Fitnessfunktion, die nach einer abgeschlossenen Suchrunde die Parameter in der Fitnessfunktion an die aktuellen Verhältnisse immer wieder neu anpassen kann. Diese und andere Themen werden ausführlich im nächsten Abschnitt behandelt, in dem neue Ansätze und Fortschritte präsentiert werden, die die grundständige Idee des PSO erneuern und erweitern. Neben einer adaptiven Fitnessfunktion werden mehrere, voneinander vollständig unabhängige Schwärme gleichzeitig in demselben Suchraum eingesetzt um verschiedene Objekte zu finden. Weiterhin wurde eine Self-Splitting-Komponente entwickelt, die die Abstände der Objekte voneinander misst und die Objekte dann als ge-

trennt oder zusammengehörig wahrnimmt. Die zwei letztgenannten Verfahren wurden noch nie zuvor realisiert.

3.1.4 Ansätze zur Realisierung eines PSO mit mehreren, autonomen Schwärmen

Es wurden mehrere Verfahren erprobt, um ein auf PSO basierendes Tracking mehrerer Objekte bzw. Personen zu verwirklichen. Hier wurden umfangreiche Tests durchgeführt, welche Farbräume oder Operatoren hilfreich sein können.

Zunächst wurde ein Video, Auflösung 576x720 Pixel, in RGB aufgenommen. Zwei Personen bewegen sich darauf, kommen aber nicht miteinander in Kontakt oder berühren sich nicht (Komplexitätsreduktion). Zheng/Meng verwenden in ihrem Artikel "'The PSO-Based Adaptive Window for People Tracking"' [8] etwa nur den Hue-Wert im HSV Farbraum. Allein dieser Wert ist an sich in mehreren Tests nicht immer zuverlässig, bei mehreren Objekten ist das Tracking auf PSO Basis mit nur diesem Farbwinkel nicht möglich. Vor allem bei sich verändernden Lichtverhältnissen wird der Hue-Wert oft problematisch.

Deshalb wurden bei Tests mehrere Farbwerte kombiniert, z.B. rot und blau aus dem RGB Farbraum und der Hue-Wert aus dem HSV Farbraum. Das Konstrukt bleibt jedoch sehr anfällig bei Lichtveränderungen, außerdem ist dann natürlich ein Vorwissen nötig, in welchen Farbspektren das zu trackende Objekt hohe Werte aufweist.

Die Umsetzung der interessanten Spektren in Fitnessfunktionen, die dem PSO als zu optimierende Funktion dienen, gestaltet sich einfacher: Ein möglichst gleichmäßig besser werdendes Ergebnis drückt den Schwarm schnell in die richtige Richtung. Dies hängt mit dem gleichmäßigen Konvergieren in einer bestimmten Richtung zusammen, siehe Tabelle Seite 29.

Dieses Konzept entspricht also der Idee, dass 2 Schwärme 2 verschiedene Objekte tracken (nur die Objekte überhaupt zu treffen, zwischen z.B. Kopf und Körper der Person kann so nicht unterschieden werden). Wie ist es nun möglich, dass sich die beiden Schwärme nicht vermischen bzw. das falsche Ziel anfliegen? Durch das Suchen in verschiedenen Fitnessfunktionen bzw. Farbräumen kann keine Störung auftreten, da die beiden Schwärme nicht dasselbe Ziel suchen. Analog zum natürlichen Vogelschwarm würde man sagen, dass beide Schwärme zwar relativ nahe beieinander fliegen, aber nicht auf der Suche nach demselben Futter sind. Vogelart 1 mag das Futter der Vo-

gelart 2 nicht und umgekehrt. Obwohl die beiden Schwärme unter Umständen relativ nahe beieinander sind, interessiert sie das Ziel des anderen Schwarms nicht. Das hat zur Folge, dass sie auch die Flugrichtungen, Geschwindigkeiten usw. des anderen Schwarms nicht wissen. Nur von ihrem eigenen Schwarm sind ihnen wie gehabt Position, Geschwindigkeit, global best usw. bekannt.

Ein weiteres Konzept geht von einer anderen Idee aus: Angenommen, es gibt 2 Schwärme in einem bestimmten, sehr großen Bereich, die dasselbe Futter suchen, zu Beginn aber sehr weit auseinander liegen. Angenommen, bei einem Bild mit Auflösung 576x720 liegt das erste gbest der beiden Schwärme 300 Pixel auseinander. Beide suchen in der nächsten Runde also dort weiter, die Geschwindigkeit jedes Partikels ist variabel, aber auf maximal 16 Pixel pro Suchrunde begrenzt. Die Wahrscheinlichkeit, dass sich ein Schwarmpartikel aus Vogelart 1 zu Vogelart 2 "'verirrt"', geht bei obigen Annahmen gegen 0 (in Tests ermittelt). Einer der beiden Schwärme wird also gezielt in ein lokales Maximum getrieben, das eigentliche globale Maximum ist viel zu weit entfernt. Beispiel aus der Natur: Vogelschwarm 1 befindet sich am Stammgelände der TUM, Vogelschwarm 2 in Garching. Vogelschwarm 1 findet in der Innenstadt einen Futtervorrat der Größe x. Vogelschwarm 2 findet in Garching eine Futterstelle der Größe $y = 3x$. Theoretisch müsste sich Schwarm 1 nun auf den Weg nach Garching machen. Da die Vögel von Schwarm 1 aber nichts von Schwarm 2 wissen und die Wahrscheinlichkeit, dass ein Vogel von Schwarm 1 zufällig nach Garching fliegt, gegen 0 geht, da die Entfernung viel zu groß ist, wird sich Schwarm 1 mit dem lokalen Maximum in der Innenstadt begnügen.

Die grundsätzliche Idee, mehrere Schwärme zu verwenden, um mehrere verschiedene Ziele zu finden und zu verfolgen ist wie der ursprüngliche PSO aus der Natur entliehen und wurde vom Autor entwickelt. In der Natur existiert eine Vielzahl von Vogelarten, die sich in Schwärmen fortbewegen. Dabei vermischen sich Vögel verschiedener Arten niemals, ihr Instinkt zeigt ihnen klar an, dass sie sich zu ihresgleichen gesellen sollen. Die Natur hat dies so eingerichtet, da Vögel einer bestimmten Gattung dieselben Eigenschaften zur Verfügung haben, dieselbe Nahrung suchen und dasselbe vorhaben, z.B. im Winter in den Süden fliegen oder nicht. Diese Eigenschaften lassen sich nun, wie beim ursprünglichen Konzept des PSO, ebenfalls in einen Algorithmus übertragen: Besitzen die Schwärme verschiedene Fitnessfunktionen, suchen sie nicht mehr dieselbe Nahrung. Die Schwärme vollständig autonom zu machen ist

ebenfalls mit einem Algorithmus lösbar: Innerhalb eines Schwarms kommunizieren alle Individuen miteinander und jedes Mitglied weiß, was der aktuell gefundene größte Nahrungsvorrat ist. Der "'fremde"' Schwarm bekommt diese Informationen nicht, besitzt innerhalb seines Schwarmes jedoch dieselbe Kommandostruktur. Beide Schwärme agieren also ähnlich, ohne sich gegenseitig zu stören.

Realisierung eines PSO mit mehreren, autonomen Schwärmen

Auf Basis dieses Ansatzes wurde die weltweit erste PSO-basierende Tracking Software erstellt, die mit zwei völlig autonomen Schwärmen arbeitet. Zwar existierte die Idee, mehrere Schwärme zu verwenden, bereits in der Vergangenheit, diese waren jedoch nicht autonom. Die bereits existierenden Realisierungen arbeiten in der Regel mit mehreren Schwärmen, die untereinander vernetzt sind. So kann ein Schwarm beispielsweise eine untergeordnete Aufgabe abarbeiten, sein Teilergebnis dann an den ihm übergeordneten Schwarm weiterreichen und von Neuem suchen [19]. Dieses Konzept wird allgemein als hierarchische Schwärme bezeichnet. Jeder Schwarm erhält so gewisse Befugnisse innerhalb des Gesamtschwarms, der in mehrere Unterschwärme gegliedert ist. Diese Realisierung hat mehrere Nachteile: Versagt ein Schwarm, versagt meist das Gesamtkonzept, da die Aufgaben oft voneinander abhängig sind. Es ist auch nicht möglich, so zwei oder mehr völlig unabhängige Objekte nachzuführen, da alle Schwärme voneinander wissen. So können sie sich vermischen oder durch einen anderen Schwarm in ihrer Nähe irritiert werden.

Die beiden gerade erläuterten Konzepte wurden jeweils unter MATLAB realisiert, der Quellcode ist im Anhang beigefügt. Es wurde vor allem Wert auf ein leicht verständliches und schnell ausführbares Programm Wert gelegt.

3.2 Adaptive Fitnessfunktion

3.2.1 Grundlagen

Eine adaptive Fitnessfunktion ist eine Funktion, die sich über der Zeit während eines Trackingvorganges ändern kann. Dies kann sinnvoll sein, wenn genügend Daten verfügbar sind, die Aufschluss über die aktuelle Situation im Bild geben. Dieser Gedanke ist nicht neu und wird auch ohne PSO mit anderen Verfahren bereits verwendet.

Etwa ist folgendes Szenario denkbar: Der PSO hat zwei Frames fertig analysiert und die darin enthaltenen Objekte nachgeführt. Der Algorithmus ermittelt aus Daten, die er in der Vergangenheit analysiert hat, dass zwei dieser Objekte sich mit hoher Geschwindigkeit aufeinander zubewegen. In einer solchen Situation ist es wahrscheinlich, dass diese beiden Objekte sich entweder weiter aufeinander zubewegen óder sogar kollidieren bzw. hintereinander vorbeifliegen, der Unterschied zwischen diesen zwei Möglichkeiten ist auf einem zweidimensionalen Bildmaterial oft nur sehr schwer zu unterscheiden. Nun können in diesen Fall Situationen entstehen, die das Tracking der beiden Objekte stark erschweren, z.B. kann ein Objekt hinter dem andern ganz oder teilweise verschwinden. Ist so eine Situation nun absehbar, schaltet eine adaptive Fitnessfunktion weitere Kriterien oder Hinweise hinzu, um die Objekte leichter zu identifizieren und zu unterscheiden.

Der bereits realisierte PSO verwendet nur die Grauwerte, um Objekte nachzuführen. Will man nun mehrere Objekte nachführen, geliegt dies mit Grauwertstufen als einziges Kriterium nur, wenn der Abstand zwischen den Objekten "'groß genug"' ist. Die notwendige Größe des Abstandes hängt dabei von vielen Faktoren wie Größe der Objekte, Größe des Videos in Pixel u.a. ab.

$$G1_{Pic2-Pic1} = \begin{pmatrix} 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 14 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 25 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 5 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 5 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 19 & 32 \end{pmatrix} \tag{3.1}$$

$$G2_{Pic2-Pic1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 17 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 5 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 19 & 32 \end{pmatrix} \tag{3.2}$$

Die beiden Gleichungen 3.1 und 3.2 sollen den Zusammenhang verdeutlichen. Beide stellen die Grauwertdifferenzen von zwei aufeinanderfolgenden Frames dar, zu Anschauungszwecken von 576x720 auf 20x10 Pixelpositionen reduziert. In Gleichung 3.1 befindet sich ein Objekt oben links und ein Objekt unten rechts, beide Objekte sind dabei relativ weit auseinander. Beide Objekte zeigen dabei konvergierendes Verhalten, an den Rändern sind die Grauwertdifferenzen kleiner als im Zentrum des Objektes. In dieser Situation liegt nun ein sehr großer Bereich zwischen den beiden Objekten, die Wahrscheinlichkeit, dass sich ein Individuum von links oben nach rechts unten "'verirrt"', ist dadurch relativ gering. Dies liegt an der Eigenschaft des PSO, nur in Richtungen weiter zu suchen, in denen hohe Werte aufgefunden wurden. Durch Begrenzung auf eine Maximalgeschwindigkeit von z.B. 5 Pixel pro Epoche pro Individuum, verspürt ein Individuum, das nur den Wert "'0"' gefunden hat, eine sehr hohe Anziehungskraft in Richtung der erfolgreicheren Komponenten. Es würde also sofort in Richtung "'seines"' Objektes zurückdriften, da in den anderen Richtungen weder große Werte noch konvergentes Verhalten zu finden ist. Umso größer dabei der Unterschied zwischen dem erfolgreichsten Partikel und dem "'verirrten"' Partikel, desto stärker wird die soziale Komponente des verirrten Individuums und damit der Drang, sich in die Richtung zu bewegen, in der es sich dem erfolgreichsten Partikel nähert. Der andere Schwarm ist dabei völlig autonom und kommuniziert mit dem anderen Schwarm in keinster Weise, kann diesen also auch nicht beeinflussen.

Gleichung 3.2 zeigt nun deutlich die Notwendigkeit einer erweiterten bzw. adaptiven Fitnessfunktion auf. Das Objekt, das sich gerade noch links oben in der Ecke

befand, hat sich auf das andere Objekt zubewegt. Der Abstand zwischen den beiden Objekten ist fast verschwunden. Es existiert wie vorher keine Möglichkeit, anhand der Grauwerte die beiden Objekte zu unterscheiden. Im Fall 3.1 war dies auch nicht nötig, durch den großen Abstand war ein "'Überspringen"' einzelner Individuen des Schwarms ausgeschlossen. Im Fall 3.2 ist dieser sichere Abstand nicht mehr gegeben. Ein Individuum könnte nun ohne Probleme in einer Epoche auf das "'falsche"' Objekt überspringen. Da die Objekte grundsätzlich so nicht unterscheidbar sind und nur nach hohen Grauwertdifferenzen gesucht wird, könnte dieses eine Individuum dann weitere Individuen auf das falsche Objekt "'herüberziehen"'. Das Objekt, das über die höchste Grauwertdifferenz verfügt, in diesem Fall das Objekt mit der Differenz 32 in der Mitte, würde stets als Sieger aus dem Optimierungsprozess hervorgehen.

Im Fall 3.1 dagegen gibt sich der Schwarm links oben stets mit dem lokalen Maximum 25 zufrieden, da das globale Maximum 32 aufgrund der nicht konvergierenden Struktur der gesamten Matrix nicht auffindbar ist. Seine Individuen sind in 3.1 zu weit entfernt.

Um nun die Fitnessfunktion optimal an diese wechselnden Aufgaben anzupassen, empfiehlt sich eine adaptive Fitnessfunktion. Im Fall 3.2 wäre eine Fitnessfunktion notwendig, die zusätzliche Kriterien, beispielsweise den Hue-Wert im HSV-Farbraum, als Unterscheidungskriterium verwendet. In Situation 3.1 dagegen würde eine Unterscheidung nach Grauwertdifferenzen, wie bereits beschrieben, ausreichen. Zwar wäre es auch möglich, für jede Situation zusätzliche Kriterien zu verwenden, allerdings würde dies einen erhöhten Rechenaufwand mit sich bringen als auch zusätzliche Anfälligkeiten, beispielsweise ist der Hue-Wert gegenüber Lichtveränderungen höchst fehleranfällig. Die zusätzlichen Kriterien sollten also nur dann verwendet werden, wenn es sich nicht vermeiden lässt.

3.2.2 Realisierung

Zur Verifizierung der gemachten Aussagen und für Testläufe wurde eine PSO-basierende Tracking Software mit adaptiver Fitnessfunktion erstellt. Sowohl zwei Personen, die gleichzeitig im Bild sind als auch zwei Objekte, getestet mit z.B. Bällen oder Handtüchern, können mit zwei verschiedenen Schwärmen nachgeführt werden.

Abbildung 3.1 zeigt eine selbstaufgenommene Sequenz. Die beiden Objekte in diesem Bild können relativ problemlos getrackt werden: Das "'global best"', also die Po-

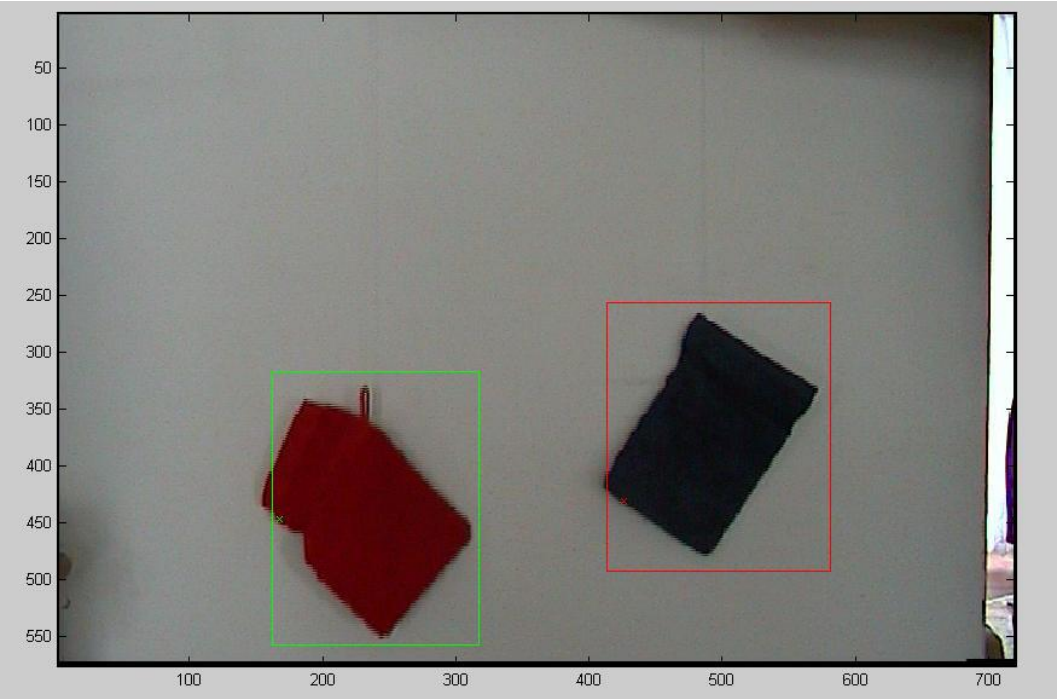

Figure 3.1: Trackingsequenz mit 2 verschiedenen Objekten und adaptiver Fitnessfunktion

sition, die das erfolgreichste Individuum des jeweiligen Schwarms gefunden hat, wird auf dem blauen Handtuch durch ein rotes Kreuz, das gbest auf dem roten Handtuch durch ein grünes Kreuz symbolisiert. Einzelnen Schatten auf dem Bild, z.B. rechts oben der sich bewegende Schatten eines Armes, gegenüber ist der PSO sehr robust: Da die Geschwindigkeit jedes Partikels pro Suchrunde limitiert und der dynamische Hintergrund viel zu weit entfernt ist, kann es die beiden Schwärme nicht irritieren. Bei einem herkömmlichen Tracking müsste man den Suchraum begrenzen, um nicht durch den Schatten irritiert zu werden, der PSO tut dies durch die Begrenzungsmasken für die Partikel gewissermaßen von selbst.

Die Lichtverhältnisse in diesem Bild sind gewollt diffus: Die Aufnahme ist sehr dunkel, ein Arm und die beiden Handtücher werfen jeweils Schatten auf das Bild. Für diese und ähnliche Situationen ist der PSO sehr robust, da der Schatten in der Regel eine niedrigere Grauwertänderung als das Objekt selbst hervorruft, da er weniger Farbtiefe besitzt. Ebenso stören sich ändernde Umgebungen nicht, solange sie sich nicht in der Nähe der Objekte befinden. Bewegt sich etwa ein drittes Objekt durch das Bild, dessen Abstand von den anderen weit genug ist, werden die Schwärme dadurch nicht gestört.

Kommen sich die beiden Objekte nun sehr nahe, sind zusätzliche Kriterien zur Unterscheidung notwendig. In diesem Fall ist der Hue-Wert sinnvoll. Ein Handtuch ist dunkelrot, das andere blau. 360 Grad entspricht in diesem Farbwinkel der Farbe rot, blau ist etwa 240 Grad. Normiert man diese Werte, könnte man z.B. definieren: Rot ist, wenn der Hue-Wert größer als 0.95, blau ist, wenn der Hue-Wert größer als 0.65 und kleiner als 0.68 [11].

Diese Versuchsreihe wurde auch mit Bällen durchgeführt: Anhand der Grauwerte können beide Bälle in diesem Fall nicht unterschieden werden, da beide Schwärme "'dieselbe Nahrung suchen"', hier hohe Grauwertänderungen. Deshalb wird der Hue-Wert zur Unterscheidung hinzugezogen, dieser Wert ist jedoch sehr lichtempfindlich als auch störanfällig in Bezug auf einzelne Ausreißer, z.B. ein einzelner rötlicher Punkt auf dem blauen Ball oder ähnliches. Der Pseudocode des Algorithmus lautet in etwa wie in Abbildung 3.2.

```
Für das Partikel an Stelle x/y lese die Grauwertänderung aus
        Überprüfe an dieser Stelle die hue-Maske
                If hue-Wert an dieser Stelle größer 0.98
                        Multipliziere Grauwert an dieser Stelle mit 3
                End
        Übertrage diesen Wert in den out bzw. Score Vektor
        Teile den Wert an dieser Stelle wieder durch 3 falls If-Schleife durchlaufen
        End
End
```

Figure 3.2: Exemplarischer Pseudocode für ein rotes Objekt

Die Entscheidung, ob das Partikel zum roten Ball gehört, wird also nicht allein durch den Hue-Wert getroffen, das wäre sonst viel zu störanfällig, z.B. bei Lichtänderungen. Primärer Entscheidungsträger bleibt der Grauwert, der passende Hue-Wert an dieser Stelle erhöht nur den Score des Grauwertes (bei Tests wurde der Grauwert z.B. mal 3 genommen). Somit wird dann z.B. ein Punkt mit Grauwertänderung 10 aber im roten Farbwinkelbereich einem Punkt mit Grauwertänderung 25 aber nicht rot vorgezogen ($3 \cdot 10 = 30$, damit höherer Score als 25). Denn der Punkt "'nicht rot aber Grauwertänderung 25"' kann genauso gut zum blauen Ball gehören, muss also niedriger gewichtet werden.

So ist es dann auch möglich, den Hue-Bereich für z.B. rot etwas großzügiger zu wählen: Er entscheidet ja nur mit, ob der aktuelle Punkt brauchbar ist, so wird eine größere Robustheit z.B. bei Helligkeitsänderungen oder schnellen Bewegungen erreicht. Aufbauend auf dieser Realisierung wird nun ein erster Self-Splitting PSO vorgestellt,

der ein völlig neues Anwendungsgebiet für den PSO erschließt und damit so komplexe Szenen nachführbar macht wie es bereits heute mit Hidden Markov Modellen und anderen Methoden möglich ist.

3.3 Self-Splitting PSO

Um einen echten "'Self-Splitting"' PSO zu erstellen, ist es nun nur noch nötig, ein gewisses Abstandsmaß zu definieren. Durch dieses Kriterium muss festgestellt werden, wie weit die Objekte voneinander entfernt sind und ob sie demnach als einzelne Objekte oder als ein zusammengehöriges Objekt behandelt werden sollen.

Zunächst muss man, abhängig von der Objektgröße, festlegen, ab wann zwei Objekte überhaupt als ein Objekt gelten könnten. Dieser Prozess ist relativ komplex. Hält z.B. ein Kind einen Ball in der Hand oder wirft es den Ball gerade, wenn sich der Ball in oder nahe der Hand des Kindes befindet?

Diese Szenen können mitunter sehr komplex sein. Zunächst wurde eine erste Self-Splitting PSO-basierende Tracking Software geschaffen, die zwei Objekte als ein Objekt detektiert, wenn der geforderte Mindestabstand zwischen den beiden global besten Positionen der jeweiligen Schwärme unterschritten wird. Die global beste Position ist dabei die Position des erfolgreichsten Partikels, das nach bis zu 2000 Epochen ermittelt worden ist. Die global beste Position nach einem kompletten Suchdurchlauf ist also von jedem Schwarm bekannt.

Dieser Algorithmus arbeitet also wie gehabt: Beide Objekte sind weit genug auseinander, ein Tracking nur anhand der Grauwerte ist möglich, da beide Schwärme weit genug auseinander sind und nicht voneinander wissen, durch das Subtrahieren von zwei Frames entsteht die übliche "'Hügellandschaft"', jeder PSO läuft zielgerichtet in "'sein"' lokales Maximum. Auch der Algorithmus mit dem zusätzlichen Hue-Wert kann angewendet werden, er hat ja nur die Eigenschaft, den Scorewert zu erhöhen, also keine negativen Auswirkungen auf das Tracking, nur die Rechenzeit wird so unnötig höher.

Im zweiten möglichen Szenario sind zwei Objekte sehr nahe beieinander, die Grauwertdifferenz reicht als alleiniges Kriterium nicht mehr aus. In dieser Situation erhöht der Hue-Wert den Score für die jeweilige Farbe, so kann ein Vermischen der beiden Objekte bzw. der Schwarmagenten verhindert werden. In dieser Situation wäre

eine starke Lichtveränderung dann problematischer als ohne die zusätzliche Farbinformation, während sie bei größerem Abstand der beiden Objekte ohne Bedeutung bleibt. Wie kann der Algorithmus nun unterscheiden, wie weit beide Objekte auseinander sind? Der Abstand der beiden global besten Positionen des jeweiligen Schwarms muss bestimmt werden:

$$a = |gbestb(x) - gbest(x)| \tag{3.3}$$

$$b = |gbestb(y) - gbest(y)| \tag{3.4}$$

$$c = \sqrt{a^2 + b^2} \tag{3.5}$$

Bei a, b und c handelt es sich um die drei Seiten eines rechtwinkligen Dreiecks. $gbestb$ und $gbest$ sind die global besten Positionen von zwei verschiedenen Schwärmen, jeweils in $x-$ und $y-$Richtung. Durch den Betrag ist es egal, welcher Ball sich wo befindet, es geht nur um den Abstand beider Objekte jeweils in $x-$ und $y-$Richtung. Mit dem Satz des Pythagoras berechnet sich dann der Abstand. Nun muss noch festgelegt werden, dass bei z.B. $c < 50$ oder $c < 30$ (dieser Wert ist natürlich von der Größe der Objekte mit abhängig) die zwei Objekte als ein Objekt erkannt werden.

Nachfolgendes Beispiel soll die funktionierende Implementierung verdeutlichen. In einer selbstgedrehten Billardszene treffen zwei Billardkugeln aufeinander. Zunächst sind sie weit genug voneinander entfernt, die Grauwertdifferenzen reichen als Unterscheidungskriterium aus. Die beiden Bälle bewegen sich aufeinander zu, die adaptive Fitnessfunktion muss die Kriterien ändern und die Farbspektren aus den Objekten extrahieren. Beide Objekte sind hier relativ gut durch den Hue-Wert unterscheidbar, ein Ball ist hellrot, der andere dunkelblau. Ab einem bestimmten Abstand wird also eine komplexere Fitnessfunktion verwendet, die auf zusätzliche Kriterien zugreifen kann, falls nötig. Die Schwierigkeit liegt bei der Implementierung vor allem darin, festzulegen, wieviele Kriterien notwendig sind und ob auch bei schlechten Bildverhältnissen die Möglichkeit besteht, richtige Schlüsse aus der vorliegenden Szene zu ziehen. Schaltet man z.B. den Hue-Wert dazu und kurz danach wird das Bild rapide dunkler, wird dieses Kriterium kaum weiterhelfen. Ähnliches passiert im RGB-Farbraum, falls die Verhältnisse nicht wenigstens näherungsweise konstant bleiben.

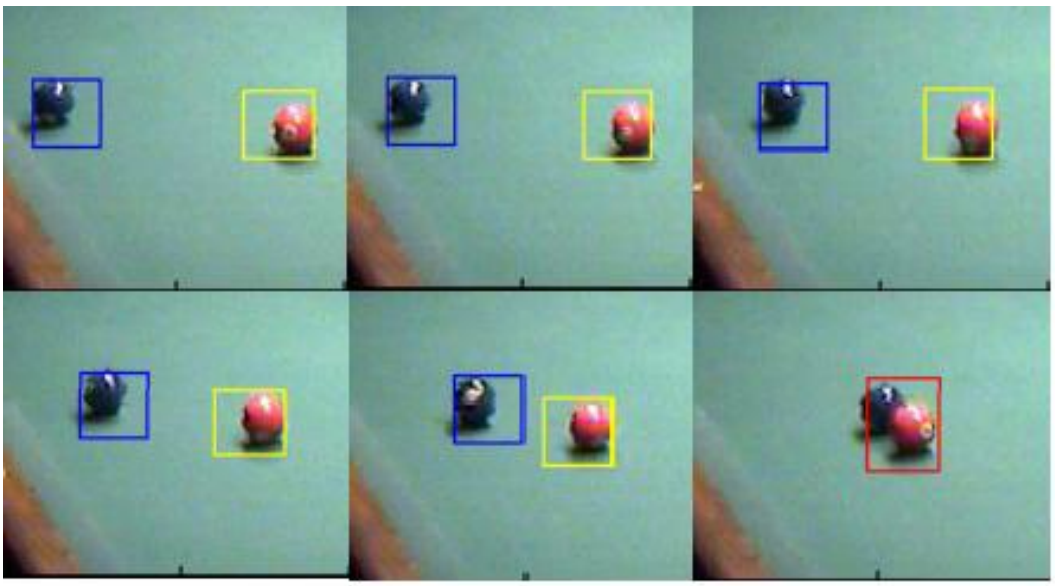

Figure 3.3: Billard-Szene mit Self-Splitting Tracking Software

Abbildung 3.3 verdeutlicht nochmals den Zusammenhang. Im letzten Teilbild wird das notwendige Mindestabstandsmaß des Algorithmus unterschritten, der blaue Ball ist seitlich leicht versetzt hinter den roten Ball gerollt. Die beiden globalen besten Positionen sind nahe genug beieinander, um als ein Objekt aufgefasst zu werden. Dennoch bleiben die Schwärme in dieser wie auch in allen anderen Situationen völlig autonom, sie wissen nichts vom jeweils anderen Schwarm. Deswegen ist es bei einer erneuten Vergrößerung des Abstandes auch problemlos möglich, beide Objekte wieder als getrennt oder alleinstehend aufzufassen.

Abbildung 3.4 zeigt nochmals zwei Objekte, hier einfach Softbälle, die sich bei diffusen, sehr dunklen Lichtverhältnissen vor einer weißen Wand bewegen. Hier werden besondere Anforderungen an den Algorithmus gestellt, vor allem die Extraktion der Farbwinkel ist hier sehr wichtig. Beide Bälle nähern sich an, werden schließlich als ein Objekt erkannt, bewegen sich danach wieder voneinander weg und werden erneut als zwei eigenständige Objekte wahrgenommen. Der minimale Abstand, der eingehalten werden muss, um die zwei Objekte als getrennt zu erkennen, muss auch hier in Abhängigkeit von der Objektgröße, der Objektgeschwindigkeit und weiteren Parametern festgelegt werden.

Der Hue-Farbwinkel wird jedoch nie als alleiniges Entscheidungskriterium verwendet wie z.B. in [8], dafür ist er einfach zu anfällig gegenüber schlechten Bildverhältnissen sowohl in Bezug auf die Lichtverhältnisse als auch in Bezug auf die Schärfe. Es

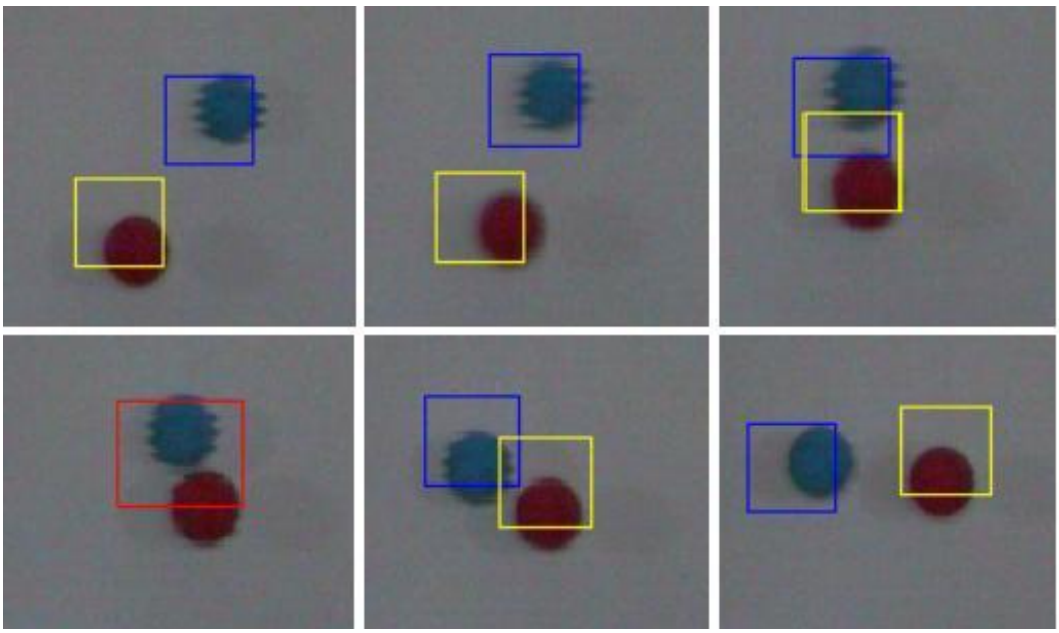

Figure 3.4: Ballspiel-Szene mit Self-Splitting Situation bei diffusen Lichtverhältnissen

wird erneut eine Abwandlung des Score-Wertes verwendet, siehe Abbildung 3.2. Als wichtigstes Entscheidungskriterium fungiert also nach wie vor der robuste Grauwert, der Farbwinkel erhöht oder erniedrigt diesen nur. Bei der Gestaltung dieser Fitnessfunktionen sind unzählige Varianten denkbar, die für den jeweiligen Fall von Vorteil sein können.

Deutlich sichtbar wird hier auch, dass der Algorithmus robust gegenüber verschiedenen Objektgeschwindigkeiten ist, solange sie nicht zu schnell werden wie z.B. beim Werfen eines Balles. Hier existiert dann die Möglichkeit, die maximale Geschwindigkeit der Partikel anzupassen. In Abbildung 3.4 ist der blaue Ball deutlich schneller unterwegs als sein rotes Pendant. Dies sieht man an den Rändern des "'ausgefransten"' blauen Balles, da die Videokamera mit dem Halbzeilenverfahren arbeitet, d.h. es werden jeweils nur die geraden bzw. ungeraden Zeilen für ein Bild verwendet, anschließend werden die beiden Halbbilder wieder zusammengesetzt. Dieses Zusammensetzen erfordert spezielle Verfahren, da sonst bei starken Bewegungen im Bild Frames entstehen können, deren Inhalt nicht mehr zusammenpasst oder verzerrt erscheint, wie hier auf dem blauen Ball sichtbar. Die beiden wichtigsten Verfahren, um Halbbilder wieder zusammenzufügen, sind "'Bob"' und "'Weave"' [11]. Insbesondere bei Weave können bei schnellen Bewegungen "'Mäusezähne"', wie hier auf dem blauen Ball zu beobachten, entstehen.

Als maximale Partikelgeschwindigkeit wurde in diesem Beispiel 16 Pixel pro Runde gewählt. Versuche haben dabei gezeigt, dass auch sehr schnelle Bewegungen verfolgt werden können, wenn man die maximale Geschwindigkeit höher wählt, z.B. 25 Pixel. Allerdings stellen sich ab einer gewissen Zahl keine Verbesserungen mehr ein, diese lag bei etwa 30, hier muss aber berücksichtigt werden, dass solche Einstellungen stets von der Auflösung des Videos abhängen. Dennoch ist die generelle Robustheit gegenüber Geschwindigkeitsänderungen bei PSO sehr beeindruckend, bei den meisten anderen Verfahren muss ein fester Suchbereich definiert werden, der meist in der Nähe des letzten Treffers lag. Bewegt sich das Objekt dann plötzlich schneller, fällt die aktuelle Objektposition möglicherweise nicht mehr in den aktuellen Suchbereich und kann damit nicht gefunden werden. Solche Unwägbarkeiten sind am besten mit einem stochastischen System, also beispielsweise PSO oder Hidden Markov Modellen realisierbar. Letztgenannte müssen jedoch eine hohe Bandbreite von Trainingssequenzen mit verschiedenen Geschwindigkeiten durchlaufen, um die gewünschte Reaktion zu zeigen.

Zuletzt bleibt bei diesem Self-Splitting zu klären, ob sich der Algorithmus theoretisch auf n Objekte ausdehnen lässt und wo praktisch die Grenze ist. Getestet wurde der Algorithmus mit bis zu drei Objekten, eine Testversion lief kurzzeitig mit vier. Je mehr Objekte dabei im Bild sind, desto höher ist natürlich der Rechenaufwand. Auf einem Standard PC heutiger Bauart ist der Self-Splitting PSO mit vier Objekten kaum echtzeitfähig, da er umso öfter zusätzliche Kriterien zuschalten muss, je mehr Objekte im Bild sind. Umso mehr Objekte im Bildbereich agieren, umso öfter wird der minimale Abstand unterschritten, wodurch die adaptive Fitnessfunktion zusätzliche Kriterien hinzuschaltet. Verwendet man nur den Hue-Wert ohne Grauwertstufen, ist dieser äußerst fehleranfällig.

Eine Änderung der Suchmaskengröße brachte kaum Veränderungen am Ergebnis. In den Testläufen wurde in der Regel eine aus drei oder sechs Pixel bestehende Suchmaske verwendet. Dazu wurde an jeder einzelnen Pixelposition der Wert ausgelesen, anschließend alle Werte zusammenaddiert.

3.4 Eigenschaften des neuen Self-Splitting PSO

Im Folgenden soll auf die Eigenschaften des neuen Self-Splitting PSO mit adaptiver Fitnessfunktion detalliert eingegangen werden. Nach generellen Beobachtungen folgen Statistiken und Daten über die einzelnen Erkennungsleistungen in verschiedenen Situationen. Allerdings sind diese mit anderen Methoden immer nur bedingt vergleichbar: ein definitiver Vergleich ist nur mit genau derselben Videosequenz möglich. Auch stellt sich beispielsweise die Frage, ab wann ein Element als richtig erkannt gewertet werden kann. Hierzu gibt es in der Fachliteratur verschiedene Ansätze, z.B. kann als richtig gelten, wenn das Objekt zu 50 Prozent vom Trackingfenster getroffen wurde. Es kann aber auch als falsch ausgelegt werden.

3.4.1 Detektion von Bewegungsrichtungen

Wie bereits erläutert gibt es bestimmte Verfahrensweisen, z.B. den Optischen Fluss, der Bewegungsrichtungen im Bild anhand der Helligkeitsgradienten feststellen und mit Vektorpfeilen markieren kann. Bei PSO ist es zwar nur schwer möglich, alle Blöcke oder Pixel verfolgen zu lassen, eine grobe generelle Richtungsangabe, wohin sich das Objekt bewegt, ist jedoch problemlos möglich.

Zunächst ist es dazu nötig, wieder zwei aufeinanderfolgende Frames voneinander zu subtrahieren. Die entstandene Grauwerttabelle weist dann an den Rändern des Objektes hohe Grauwerte auf. Will man nun nur die Werte haben, die auf der Seite der Hauptbewegungsrichtung entstanden sind, kann man z.B. die negativen Grauwerte auf der anderen Seite einfach vernachlässigen oder auf den Wert null setzen. So bekommt man starke Grauwertänderungen in der Hauptbewegungsrichtung. Allerdings funktioniert dieser Ansatz nur unter der Voraussetzung, dass sich das Objekt sehr deutlich vom Hintergrund unterscheidet.

$$Gx_{Pic2-Pic1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -34 & -12 & -3 & 1 & 0 & 0 & 13 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -36 & -11 & -4 & 1 & 0 & 2 & 20 & 28 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -24 & -14 & -4 & 0 & 3 & 0 & 12 & 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -34 & -13 & -5 & -4 & 0 & 0 & 8 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -45 & -16 & -6 & 0 & 0 & 2 & 9 & 27 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.6}$$

$$Gx2_{Pic2-Pic1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 13 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 20 & 28 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 12 & 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 9 & 27 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.7}$$

Die Matrizen 3.6 und 3.7 sollen die Idee stark vereinfacht verdeutlichen: In 3.6 sieht man, dass sich das Objekt vermutlich nach rechts bewegt hat. Am linken Rand sind negative Grauwertdifferenzen auszumachen, d.h. hier waren ein Frame vorher vermehrt dunkle Stellen, die dann heller geworden sind. Auf der rechten Seite dagegen sind die Werte positiv, d.h. hier wurden helle Stellen dunkler.

Um aus diesen Informationen nun Rückschlüsse auf Bewegungsrichtungen zu ziehen, ist einiges Wissen über das Bildmaterial vonnöten. In diesem Beispiel bewegt sich ein Ball vor einem weißen Hintergrund. Dies kann man eventuell durch genauere Analyse des Hintergrundes herausfinden, dies ist aber abhängig von der Qualität des Bildmaterials. In 3.7 sieht man nun die Grauwertänderungen, die die Hauptbewe-

gungsrichtung anzeigen. Die meisten Gradienten im Bild würden also nach "'rechts"' zeigen.

Ist das Bild nun auf diese Weise vorbereitet (in Matlab ist dies relativ einfach, da negative Werte standardmäßig eliminiert werden), kann mit PSO die Hauptbewegungsrichtung extrahiert werden. Hierzu gibt es verschiedene Möglichkeiten, die erste besteht einfach darin, nur nach dem globalen Maximum zu suchen, in diesen Fall wäre das der Wert 28. Dieses Verfahren ist allerdings recht ungenau, im Beispiel würde dann die Bewegungsrichtung nach rechts oben zeigen, obwohl bei der Gesamtbetrachtung die Bewegungsrichtung eher nach rechts geht. Die anspruchsvollere, aber auch rechenaufwändigere Methode ist es, verschiedene Helligkeitsgradienten mit der Fitnessfunktion auszuwerten. Zunächst muss die Länge der möglichen Gradienten festgelegt werden. Wählt man den Gradienten zu kurz, ist das Ergebnis eventuell schneller gefunden aber auch ungenauer. Wählt man den Gradienten sehr lang, ist auch der Suchprozess sehr lang und unter Umständen nicht erfolgreich.

Exemplarisch könnte der Gradient genau drei Pixelpositionen lang sein. An Position $x = 12, y = 6$ findet sich der Wert 12. Mit anderen Suchmethoden würde man nun z.B. alle acht Nachbarpixel überprüfen und den größten Wert, hier die 28, wählen. Anschließend würde man von dieser Position $x = 13, y = 7$ wieder auf Suche gehen und den Wert 23 auswählen, schon hätte man einen Gradienten der Länge drei. Bei PSO gestaltet sich diese Idee nun weit schwieriger, da zwar jedes Individuum Ergebnisse suchen kann, jedoch nicht auf gefundenen Teilergebnissen von anderen Individuen aufbauen kann. Es ist also nicht möglich, dass Partikel eins die erste, Partikel zwei die zweite Stelle des Gradienten findet usw. Dieses Handeln würde einer ständigen Änderung der Fitnessfunktion gleichkommen. Dies ist mit der adaptiven Fitnessfunktion zwar möglich, jedoch nur nach einer kompletten Suchrunde. Ein Verändern nach jeder Epoche ist unmöglich: So würde sich die Wertigkeit der einzelnen Positionen während des Suchprozesses ändern, das Konzept des PSO wäre konterkariert.

Die einzige Möglichkeit besteht mit PSO darin, komplette Gradienten mit der Fitnessfunktion zu suchen. Also beispielsweise überprüft ein Partikel eine Position, untersucht gleichzeitig eine direkte Nachbarposition und vergibt dann einen bestimmten Score-Wert. Ist der Anstieg z.B. sehr groß, ist der Score-Wert auch sehr groß. Diese Methode wurde jedoch nur ansatzweise getestet, es ist fraglich ob sie wesentliche Vorteile gegenüber anderen Methoden, z.B. dem Optischen Fluss, bietet.

Die erste Methode, einfach das Maximum zu verwenden, wurde jedoch umfangreich getestet. Zwar kann die Richtungsangabe manchmal etwas ungenau sein, dennoch ist sie möglich und mit minimalem Rechenaufwand erreichbar.

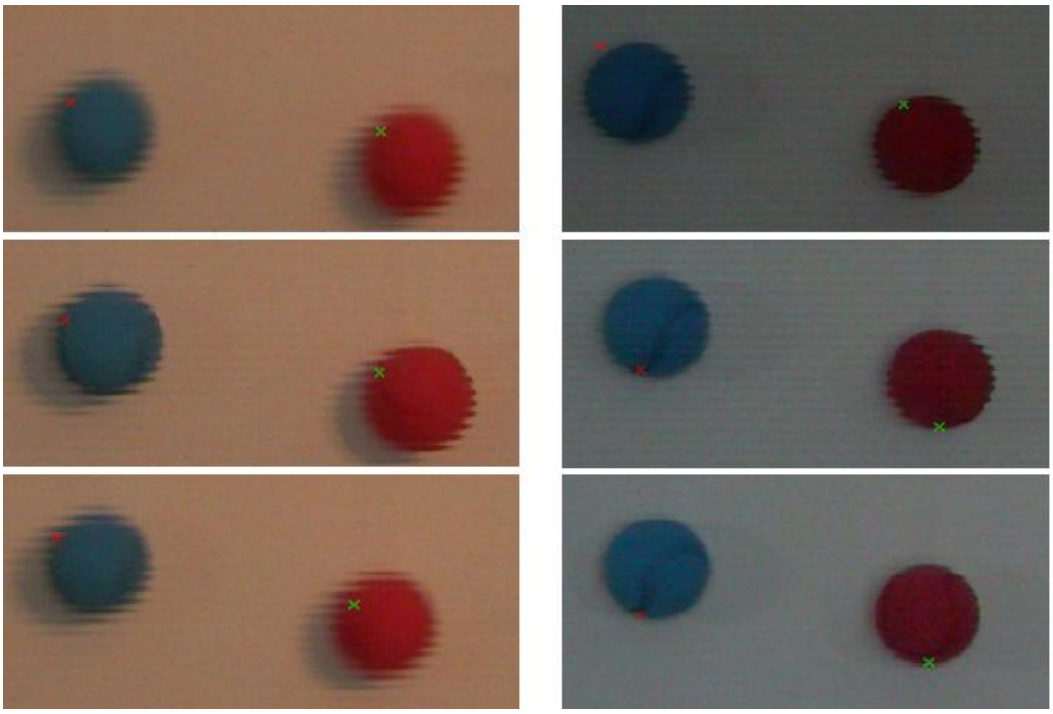

Figure 3.5: Szene mit Richtungsangabe und extremem Lichtwechsel

Abbildung 3.5 zeigt ein Beispiel, das mit PSO und der besprochenen einfachen Richtungsangabe, d.h. ohne Gradient, erzielt wurde. Die jeweiligen Bewegungsrichtungen der Bälle sind mit jeweils einem Kreuz angezeigt. Auf Bild vier tritt ein starker Helligkeitswechsel auf, der PSO ist dagegen jedoch sehr robust, er verliert weder eines der Objekte, noch verliert er die Richtungsangabe, siehe auch das zugehörige Video "'lightchangex.avi"'.

3.4.2 Hohe Robustheit

Gegenüber anderen Methoden kann der Algorithmus bezüglich Geschwindigkeits-, Licht- und Schärfeveränderungen als sehr robust gelten, dies wird im Folgenden noch mit Simulationsdaten verifiziert. Ein gutes Beispiel ist in Abbildung 3.5 sichtbar, die Bälle ändern bei starker Lichtveränderung ihre Bewegungsrichtung und ihre Geschwindigkeit. Dennoch bleiben die Kreuze zuverlässig an den richtigen Positionen.

In diesem Beispiel wurden keine zusätzlichen Kriterien wie etwa Farbwerte verwendet, ausschließlich die Grauwertdifferenzen zeigen die Richtung an.

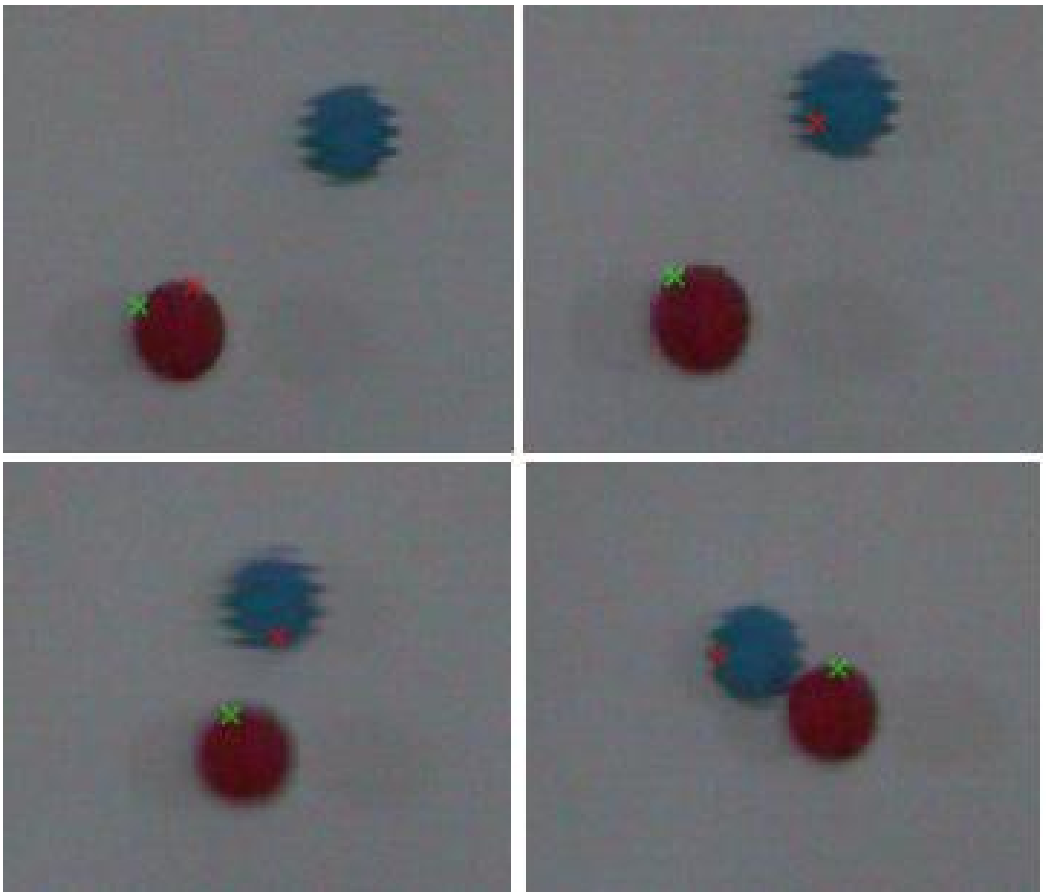

Figure 3.6: Falsche Initialisierung

Abbildung 3.6 zeigt eine bemerkenswerte Eigenschaft des PSO. Wie bereits zuvor erläutert, müssen die Objekte in der ersten Runde mit anderen Methoden gesucht werden, z.B. Kantendetektion und Extraktion der hohen Grauwertänderungen. Im Beispiel 3.6 ist diese Initialisierung nun schiefgegangen, beide Kreuze befinden sich auf dem roten Ball. Der PSO bekommt nun diese Daten übermittelt. Da bei so einer Ausgangslage bei einer funktionierenden Nachführung beide Objekte sehr nahe beieinander wären, da ja beide global bests bzw. die beiden Kreuze sehr nahe beieinander sind, schaltet der PSO sofort auf eine erweiterte Fitnessfunktion, die zusätzlich den Farbwinkel verwendet. So ist es dann möglich den Fehler der Initialrunde wieder gut zu machen. Dieses hat folgenden Hintergrund: Für alle 20 Individuen des Schwarms muss eine Startposition gefunden werden. Wird in der Initialrunde nun das falsche Objekt erwischt, befinden sich zwar einige Partikel auf dem falschen Objekt, einige aber auch auf dem richtigen.

Verbleiben auch nur ein oder zwei Partikel auf dem "'richtigen"' Objekt, werden die übrigen 18 oder 19 sehr schnell durch diese ein oder zwei Partikel und ihre hohen Score-Werte aufgrund des Farbwinkels in die richtige Richtung gezwungen. Die übrigen 18 oder 19 Partikel merken, dass weiter entfernt von ihnen viel größere "'Nahrungsvorräte"' zu finden sind. Ihr eigenes Objekt ist rot und damit nicht in dem von ihnen gesuchten Farbbereich. Bei den meisten anderen Tracking-Algorithmen besteht nach einer falschen Initialisierung meist nur eine sehr geringe Chance, das richtige Objekt wiederzufinden. Die Stärke des PSO liegt hier wieder in seiner Masse, man kann 20 oder 30 Stellen anfliegen und die Wahrscheinlichkeit einen guten Treffer zu landen ist somit höher als bei nur einer Stelle.

Als relativ robust darf der Algorithmus auch in Bezug auf kurzzeitiges Verschwinden des Objektes gelten. In Abbildung 3.3 rollen die Billardkugeln nahezu hintereinander vorbei. Besonders ein partielles Verschwinden des Objektes stört die Nachführung nicht, da ja die Form des Objektes nie als Kriterium verwendet wird. Zwar wäre dies möglich, zöge aber enormen Programmieraufwand nach sich, da für jede Form dann eine neue Fitnessfunktion geschaffen werden müsste. Zudem wäre bei schlechter Sichtbarkeit oder schneller Bewegung des Objektes die Form starken Schwankungen unterworfen, z.B. wären die genannten "'Mäusezähne"' bei Interlaced-Videomaterial sehr störend. Ebenfalls getestet wurde das Verschwinden des Objektes aus dem Bild. Ein teilweises Verschwinden zieht dabei keine Probleme nach sich, ein komplettes Verschwinden, insbesondere über längere Zeiträume, ist dagegen schwer in den Griff zu bekommen. Nachdem das ganze Objekt aus dem Bild verschwunden ist, findet der PSO keine Werte mehr, d.h. theoretisch müsste er an dieser Position bleiben. Praktisch jedoch ist jedes Bild von Rauschsignalen überlagert, die zwar teilweise, sehr selten aber ganz aus dem Bildmaterial herausgefiltert werden können. Somit springt die global beste Position dann auf z.B. eine geringe Grauwertänderung von drei oder vier Stufen, die jedoch durch Rauschen verursacht ist. Das Objekt kann in solchen Fällen also kaum mehr verfolgt werden, dieses Problem haben jedoch nahezu alle Tracking-Verfahren gemeinsam.

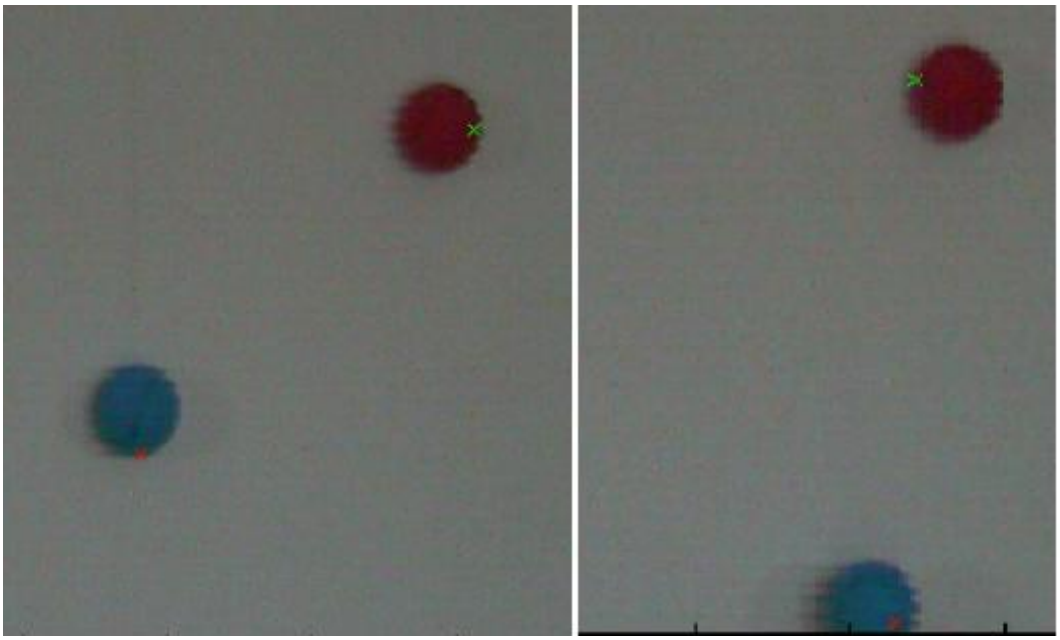

Figure 3.7: Partielles Verschwinden eines Objektes

Chapter 4

Ergebnisse und Vergleich mit anderen Methoden

Es wurde eine neuartige PSO-basierende Tracking Software geschaffen, die es erstmals ermöglicht, mehrere verschiedene Objekte unabhängig voneinander nachzuführen. Zusätzlich werden adaptive Fitnessfunktionen eingesetzt, um in verschiedenen Situationen die Objekte nicht zu verlieren. Zunächst werden im Folgenden Vergleiche mit anderen Methoden und Forschungsergebnissen gezogen, um die neuen Ergebnisse einordnen zu können. Neben einem Vergleich mit weiteren PSO-basierenden Methoden erfolgt auch ein Vergleich mit auf Hidden-Markov-Modellen basierenden Methoden.

4.1 Vergleich mit anderen PSO-basierenden Methoden

Für diesen Vergleich wurden einige andere Forschungsergebnisse auf ihre Möglichkeiten überprüft und genauer untersucht. Diese Forschungen spiegeln den aktuellen Stand der Technik wieder.

In [8] findet sich ein aktuelles Beispiel für eine PSO-basierende Tracking Software. Der Algorithmus basiert auf dem originalen Formelwerk nach Kennedy/Eberhart, vergleiche Formeln 2.1 und 2.2. Die Autoren verwenden einen fünfdimensionalen PSO, der neben der x- und y-Koordinate des Mittelpunktes des gesuchten Objektes die Breite und Höhe des Suchfensters und dessen Verdrehung gegenüber der x-Achse in Grad feststellt. Bei dieser Implementierung handelt es sich um einen State-of-the-Art PSO, es wird nur dieses eine Objekt verfolgt, die Nachführung eines weiteren Objektes ist nicht möglich wie mit dem hier vorgestellten Self-Splitting PSO. Darüber hinaus wird

als einziges Kriterium der Hue-Wert aus dem HSV-Farbraum verwendet. Somit kann der Algorithmus von [8] nur unter absolut konstanten Helligkeitsbedingungen verwendet werden, etwa in einem Büro ohne Fenster mit festen Lichtquellen. Auch darf das nachzuführende Objekt dann keine Lichtquelle verdecken oder verändern. Diese Schwachstelle wird von [8] auch nicht bestritten. Das Problem dieser Implementierung ist vermutlich der hohe Rechenaufwand für den fünfdimensionalen Suchraum, der z.B. auch spezielle Operationen wie das Größer- oder Kleinerwerden der nachzuführenden Person oder des Objektes beherrschen soll. Durch das aufwändige Suchprozedere muss der Aufwand bei den Kriterien niedrig gehalten werden. Der in dieser Arbeit vorgestellte PSO besitzt einen anderen Ansatz, statt dem fünfdimensionalen Suchraum werden adaptive Fitnessfunktionen mit wechselnden Kriterien und dafür nur ein zweidimensionaler Suchraum verwendet.

Was hier sinnvoller ist, ist abhängig von der Anwendung. Da [8] nur eine Person nachführen kann, nimmt diese unter Umständen mehr Platz im Bild ein als wenn sich drei oder mehr Objekte im Bildbereich aufhalten. Deshalb ist das "'Zoomen"' der Person auch nötiger als bei mehreren Objekten, die dann weiter von der Kamera entfernt sind und bei denen eine Größenänderung nur in einem bestimmten Rahmen überhaupt möglich ist. Weiterhin arbeitet diese Implementierung mit einer maximal zulässigen Abweichung von 15 Pixel vom letzten Mittelpunkt bzw. Treffer. Dieses enge Kriterium lässt keinerlei Spielraum für schnelle oder ruckartige Bewegungen. Im Gegensatz dazu setzt der neue PSO auf die automatische Begrenzung durch die maximale Individuengeschwindigkeit, vermutlich benötigen Zheng/Meng diese enge Grenze, um weitere Rechenkraft zu sparen. Weiterhin ist es bei [8] nötig, dem Algorithmus manuell das Objekt zu zeigen, indem man mit der Maus das Objekt mit einem Rechteck markiert. Dies stellt eine große Schwäche dar, da der Algorithmus dann nicht autonom arbeiten kann, was das eigentliche Ziel der meisten Nachführungsvorgänge ist.

Eine interessantere, von Matthias Scheutz [19] veröffentlichte Implementierung verwendet die Grauwerte als wichtigstes Kriterium. Das Verfahren von Scheutz ist dabei durchaus ambitioniert: Unter Echtzeitbedingungen sollen mit einem hierarchischen Schwarm die beiden Augen und der Mund eines Gesichtes nachgeführt werden. Das Konzept des hierarchischen Schwarms existiert bereits seit längerer Zeit, hat mit dem hier vorgestellten Self-Splitting PSO jedoch relativ wenig gemein. In einem hierarchischen Schwarm existiert meist ein übergeordneter und mehrere untergeordnete

Schwärme. Bewegt sich das Gesicht nun, sucht ein Schwarm z.B. die Augen und ein anderer Schwarm den Mund. Diese Schwärme sind aber nicht autonom, sondern sie müssen ihr Teilergebnis an einen übergeordneten Schwarm abgeben, der dann damit das Gesamtergebnis bildet [19]. Somit wissen diese Schwärme von der Existenz der anderen Schwärme und können sich auch gegenseitig beeinflussen. Scheutz definiert nun relativ komplexe Abstandsmaße: Die Augen haben einen bestimmten Abstand voneinander, der Mund ebenfalls von jedem Auge. Diese Idee hat starke Nachteile beim Drehen des Kopfes oder Neigen, da sich hier die Abstände gravierend verändern können. Es sollte vor allem gezeigt werden, dass PSO-basierende Verfahren weniger anfällig gegenüber Lichtveränderungen sind als andere Methoden (außer der Hue-Wert ist das alleinige Unterscheidungskriterium, siehe Zheng/Meng). Diese Beobachtung wurde bei den hier getätigten Versuchen ebenfalls gemacht.

Der Nachteil von diesen hierarchischen Schwärmen gegenüber dem Self-Splitting PSO ist, dass zwar mehrere Objekte verfolgt werden können, diese dürfen jedoch nicht autonom sein. Ihr Abstand und ihr Aussehen muss in etwa konstant bleiben, bei einem Gesicht, das sich nicht zu stark bewegt, ist dies gegeben. Voneinander unabhängige Objekte können mit hierarchischen Schwärmen dagegen nicht verfolgt werden.

Scheutz verwendet dabei keine adaptive Fitnessfunktion sondern stets nur die Grauwerte. Er benutzt zusätzlich eine Maske, die die Form des Objektes extrahiert. Diese Idee schränkt die allgemeine Verwendbarkeit des Algorithmus natürlich stark ein, so können etwa keine Objekte mit derselben Form (z.B. zwei Bälle) verfolgt werden, außerdem wäre es nötig, für jede neue Form einen neuen Algorithmus zu schreiben. Dies entspräche dann einem ähnlichen Konzept wie der adaptiven Fitnessfunktion. Jedoch scheint es schwer realisierbar, eine Formänderung durch einen Algorithmus zu detektieren. Scheutz verwendet keine adaptive, sondern eine festgelegte Fitnessfunktion, die nur für diese Gesichtserkennung anwendbar ist.

Zur Verifizierung der Funktion seines Algorithmus definiert Scheutz ein eigenes Fehlermaß. Um jedes zu findende Objekt zeichnet er eine Box, die den optimalen Treffer des Objektes darstellt. Dann misst er die Abweichung des Tracking-Fensters in jedem Frame und summiert die Fehler auf. Die Erkennungsleistungen werden in Tabelle 4.1 zusammengefasst [19].

Object	static	swarm	hierarchy
left	2608	1774	2214
right	3929	2082	2551
mouth	4356	3379	2896
total	10894	7237	7661

Table 4.1: Ergebnisse von Matthias Scheutz

Der Fehler ist also bei der rein schwarm-basierenden Methode deutlich niediger als bei dem hierarchischen Schwarm, am schlechtesten funktioniert, wie bei starker Lichtänderung zu erwarten, die statische Methode mit Kantendetektion.

Eine weitere interessante Implementierung wurde von Pugh/Martinelli durchgeführt [20]. Hier dient das Konzept PSO zur Auffindung und Navigation von mehreren Robotern, die sich durch einen Suchraum bewegen. Pugh/Martinelli legen dabei eine sogenannte "'Communication Range"' fest, einen Abstand, in dem die Partikel miteinander kommunizieren können. So werden von den Partikeln keine Meinungen von Individuen miteinbezogen, die sehr weit von ihnen entfernt sind. Hier ist dann unscharf definiert, ob diese Partikel de facto zu einem Schwarm gehören, da im ursprünglichen PSO-Konzept alle Partikel untereinander kommunizieren können. Beim neuen Self-Splitting PSO kommunizieren nur die Partikel miteinander, die zu einem Schwarm gehören. Angehörige eines "'fremden"' Schwarmes nehmen diese Partikel nicht wahr und kommunizieren auch nicht mit ihnen, stören sich deshalb auch nicht an ihnen.

4.2 Vergleich mit HMM-basierenden Methoden

Hidden Markov Modelle eignen sich auch sehr gut für Bildverarbeitungszwecke, ein Beispiel wird etwa in [13] gegeben. Zunächst benötigt jedes Hidden Markov Modell eine sinnvolle Zahl von Zuständen und Verbindungen in der sogenannten Markov-Kette. Dabei handelt es sich um ein stochastisches Modell, d.h. es existiert eine Wahrscheinlichkeit dafür, dass ein bestimmter Zustand aktiv ist bzw. dass von einem in einen anderen Zustand gewechselt wird. Diese Gewichtungen bzw. Wahrscheinlichkeiten müssen nun auf eine reale Anwendung, z.B. auf das Nachführen eines Objektes in einem Video, trainiert werden. In [13] werden dafür verschiedene Möglichkeiten

Sequenz	Torso	Arm	Leg
Baseball	98.4	93.75	95.3
Skating	99.2	77.5	97.6
Lola	95.6	89.8	92.6
Park	79.1	29.2	58.3
Walk	99.3	99.3	98.0

Table 4.2: Ergebnisse mit HMM-basierendem Tracker (in Prozent)

aufgeführt, z.B. die Schaffung eines Templates, also eines Musters, das eine bestimmte Situation oder Pose darstellt. Weitere Möglichkeiten sind das Konstruieren eines Likelihood-Modells, das in der Statistik ein parametrisches Schätzverfahren darstellt, also ebenfalls auf Stochastik beruht oder das Erstellen eines dynamischen Modells, das sich an wechselnde Situationen anpassen kann [13].

Diese Ansätze haben nun alle gemein, dass ein hoher Trainingsaufwand nötig ist, um mit diesem Modell gute Ergebnisse zu erzielen. Durch das Training werden die Eintrittswahrscheinlichkeiten einzelner Zustände stetig angepasst, bis sie sich schließlich kaum mehr verändern und damit als einsatzbereit gelten (sind die Veränderungen zu gering, wird dies durch einen Schwellwertfilter gemessen und das Üben abgebrochen). In [13] wurde das Modell anschließend mit verschiedenen Videosequenzen getestet, die einen Überblick über die Erkennungsleistungen liefern.

Diese Szenen wurden jeweils nach ihrem Inhalt benannt, "'Baseball"' zeigt spielende Personen, Skating ebenso (Tabelle 4.2). Lola ist eine Szene aus dem Kinofilm "'Lola rennt"' von Tom Tykwer, "'Park"' ist eine Szene in freier Natur und "'Walk"' eine simple Gangbewegung. In jeder Sequenz wurden dabei verschiedene Körperteile einer Person nachgeführt. Es handelt sich jeweils um eine Prozentangabe, die aussagt, wieviele Prozent der Versuchsreihe korrekt lokalisiert wurden. Besonders interessant ist hierbei die Szene "'Park"', die in freier Natur aufgenommen wurde und deshalb starke Lichtschwankungen aufweist. Hier lässt die Erkennungsleistung rapide nach, da das aktuelle Geschehen unter Umständen nicht mehr mit den Mustern, die gelernt wurden, vergleichbar ist. Bei PSO-basierenden Systemen herrscht hier eine höhere Robustheit gegenüber Helligkeitsschwankungen. Allerdings wurden in den getesteten Beispielen keine einzelnen Körperteile nachgeführt, sondern ganze Objekte wie etwa Bälle oder Billardkugeln. Um eine echte Vergleichbarkeit zu gewährleisten, wäre es notwendig,

Sequenz	Objekt 1	Objekt 2
Billard	97.0	93.0
Bälle	94.0	92.0
Walk around	95.0	98.0

Table 4.3: Ergebnisse mit adaptivem Self-Splitting PSO

genau dieselben Videos zu verwenden. Dennoch seien hier zum Vergleich Ergebnisse aufgeführt, die mit dem entstandenen Self-Splitting PSO mit adaptiver Fitnessfunktion erzielt wurden (Tabelle 4.3).

Bei den beiden Objekten handelt es sich um die bereits bekannten zwei Bälle, einer rot, einer blau. Hier wurden jeweils 100 Testläufe pro Szene durchgeführt, eine Nachführung wurde dabei als richtig gewertet, wenn mindestens 30 Prozent des Objektes vom Trackingfenster getroffen werden. Hier erschweren natürlich zusätzliche Feinheiten die Vergleichbarkeit: Es gibt unterschiedliche Auffassungen darüber, wann ein Objekt als "'richtig"' erkannt gezählt wird und wann nicht, auch sind prozentuale Angaben nicht immer leicht feststellbar. Nur eine sehr genaue Auswertung, etwa müssten die einzelnen Pixel gezählt werden, kann exakte Vergleichbarkeit ermöglichen. In obigem Beispiel wurde die Bewertung nach Augenmaß vorgenommen, stichprobenartig wurden auch einzelne Pixelblöcke geprüft.

Des Weiteren kann es auch in bestimmten Situationen gar nicht möglich sein, ein Objekt richtig zu erkennen, etwa wenn das Objekt von einem anderen Objekt teilweise oder ganz verdeckt wird oder kurzzeitig aus dem Bild verschwindet. [13] haben in der Szene "'Park"' ansatzweise versucht, eine bestimmte Person aus einer größeren Menschenmenge nachzuführen. Dies ist sowohl mit PSO- als auch HMM-basierenden Methoden schwierig. Die Schwierigkeit bei PSO besteht darin, dass dann jedes Objekt oder jede Person mit einer eigenen Fitnessfunktion nachgeführt werden muss, die auch genug Unterscheidungskraft im Vergleich mit den anderen Fitnessfunktionen mitbringt. Sind etwa fünf Objekte sehr nahe beieinander und sehen kaum unterschiedlich aus, ist die Nachführung sowohl mit PSO- als auch HMM-Methoden kaum lösbar.

Der Vorteil von [13] liegt vor allem darin, dass sehr viele Positionen oder Möglichkeiten trainiert werden können, was jedoch auch mit enormem Aufwand verbunden ist. Dieser Ansatz entspricht in etwa, dass man einem PSO-basierenden System sehr viele Fitnessfunktionen "'mitgibt"', aus denen dann für alle denkbaren Be-

wegungen eine passende ausgewählt werden könnte. Dies ist aber ebenfalls mit hohem Aufwand verbunden. Hier muss immer zwischen Aufwand und Situation, die bewältigt werden soll, unterschieden werden.

Der PSO-Ansatz verfügt über deutliche Vorteile bei starken Lichtwechseln (siehe Abbildung 3.5) und schnellen Bewegungen. Das HMM muss schnelle und abrupte Bewegungen erst lernen, bei PSO kann durch die Individuengeschwindigkeit bereits ein großer Teil der Situationen abgedeckt werden. Da bei der vorliegenden PSO-Implementierung in der Regel auf Formen bzw. Muster verzichtet wird, liegt auch eine hohe Robustheit bei schlechten Bildverhältnissen oder ruckeligem Bildmaterial vor.

4.3 Vergleich mit weiteren Methoden

Neben den beiden genannten, heute meistverwendeten Verfahren, gibt es eine Serie weiterer Möglichkeiten. Diese wurden bereits im Kapitel zwei besprochen, hier soll noch ein kurzer Vergleich dieser Methoden mit dem neuen Self-Splitting PSO stattfinden.

Bei der klassischen Kantendetektion fällt in komplexeren Situationen die Leistung schnell ab. Beispielsweise bei starken Lichtveränderungen ist mit dem statischen Schwellwert für Grauwerte oder Farben wenig auszurichten, da sich diese Werte dann sehr weit verschieben können. Ein weiteres Problem sind "'verschwimmende"' Kanten, wenn eine Bewegung sehr schnell oder sehr plötzlich auftritt. Die Kante wird durch "'ausfransende"' Ränder dann künstlich geglättet. Durch eine Vorverarbeitung wie z.B. ein Mittelwertfilter zur Rauschunterdrückung werden die Kanten im Bild ebenfalls verändert und sind dann unter Umständen schwerer zu extrahieren. Insgesamt stellt die Kantendetektion ohne weitere Hilfsmittel damit eher ein veraltetes Konzept dar. Zum Vergleich mit dem entstandenen PSO wurden dieselben Videos mit klassischer Kantendetektion mit Sobelfilter nachgeführt (jeweils 10 Durchläufe,siehe Tabelle 4.4).

Vor allem das Video mit starken Lichtschwankungen (Abbildung 3.5) macht dem Detektor schwer zu schaffen, der statische Filter kann nach einem Lichtwechsel meistens dem Objekt nicht mehr folgen. Generell handelt es sich um eine verwendbare Methode, falls die Aufnahme nicht zu komplex ist.

Sequenz	Objekt 1	Objekt 2
Billard	7 aus 10	8 aus 10
Bälle	4 aus 10	2 aus 10

Table 4.4: Nachführung mit Kantendetektion

Der Optische Fluss richtet sich nach den Helligkeitsgradienten im Bild. Gegenüber dem Self-Splitting PSO mit adaptiver Fitnessfunktion bekommt dieses Konzept bei starken Helligkeitsschwankungen ebenfalls Probleme, da die Gradienten dann ihr Aussehen stark verändern. Da das Konzept des Optischen Flusses Helligkeitsgradienten an anderer Stelle wieder sucht, sind auch schnelle oder abrupte Bewegungen oft schwer verfolgbar, da der Suchraum dann zu klein oder falsch gewählt sein kann. Dies ist natürlich abhängig davon, ob Blöcke bestimmter Größe oder definitiv jedes einzelne Pixel bewegungsmäßig erfasst werden soll. Verwendet man hier größere Blöcke, lassen sich auch bessere Ergebnisse erzielen.

Im Vergleich mit dem Self-Splitting PSO besitzt der Optische Fluss leichte Nachteile in Bezug auf starke Helligkeitsänderungen. Bei sehr schnellen Bewegungen ist der Erfolg abhängig von der Auslegung: Für den Optischen Fluss muss ein Suchbereich definiert werden und meist ein Verfahren, nach welchem System gesucht wird. Bei PSO kann man die Größe des Suchbereiches nur indirekt über eine Maximalgeschwindigkeit für die einzelnen Individuen festlegen.

Chapter 5

Zusammenfassung

In dieser Arbeit wurde das Konzept und die Idee, die hinter der 1995 von Kennedy und Eberhart entwickelten "'Particle Swarm Optimization"' steht, für Tracking-Anwendungen durch umfangreiche Erweiterungen erst wirklich nutzbar gemacht. Bisher war es zwar möglich, mehrere Objekte getrennt voneinander nachzuführen. In der Regel wurden dazu jedoch "'hierarchische Schwärme"' benutzt, ein Notkonstrukt, das für voneinander unabhängige Objekte oft problematisch war, da in solchen hierarchischen Schwärmen Kommunikation zwischen den einzelnen Schwärmen stattfinden muss. So beeinflussen sich die Schwärme gegenseitig und können sich gegenseitig stören. Mit dem neuentwickelten Self-Splitting PSO mit adaptiver Fitnessfunktion ergeben sich neue Möglichkeiten im Bereich Bildnachführung, die mit PSO bislang nicht möglich waren. Hierzu zählt das Nachführen von n voneinander unabhängigen Objekten im Bild, sehr gute Ergebnisse bei starker Veränderung der Lichtbedingungen, variable Reaktionsmöglichkeiten auf extreme Situationen wie z.B. sehr schnelle Bewegungen und der adaptive Charakter der Fitnessfunktion. Mit der grundlegenden Analyse des Schwarmverhaltens bei mehreren unabhängigen Schwärmen und den darauf aufbauenden Konzepten ist es erstmals möglich, hochleistungsfähige Tracking-Software zu erstellen, die zuvor in der Regel mit Hidden Markov Modellen realisiert wurden.

Mit den gemachten Fortschritten im Bereich PSO erscheint dieses Verfahren für die Zukunft das erfolgversprechendste zu sein. Auch im Bereich der Hidden-Markov-Modelle sind enorme Fortschritte gemacht worden. Aus Sicht des Autors hat das Grundkonzept der HM-Modelle entscheidende Nachteile: ein enormer Trainingsaufwand ist nötig, um brauchbare Ergebnisse zu erzielen, dies muss stets im voraus bewerkstelligt werden. Die Ergebnisse des Modells sind dabei kaum nachvollziehbar,

trifft ein HMM etwa eine Fehlentscheidung, ist nicht feststellbar, wie dieser Fehler zustande kam, da die Wahrscheinlichkeiten der Zustände und Übergänge zwar sichtbar sind, der Weg der Entscheidungsfindung aber oft unklar bleibt. Bei PSO ist zumindest die global beste Position in jeder Suchrunde feststellbar, es ist also sichtbar wie der Algorithmus gleichmäßig auf ein bestimmtes Ergebnis konvergiert. Falls Fehler auftreten ist sichtbar, wo und wann der Algorithmus das Objekt verloren hat. Es ist auch kein Trainingsaufwand nötig, PSO-basierende Systeme trainieren sich gewissermaßen während des Suchvorganges, indem sie durch ihre Masse sehr viele Lösungen nach dem bekannten Mechanismus "'durchprobieren"'.

Bei beiden Konzepten handelt es sich um stochastische Verfahren. HM-Modelle geben für jedes Ergebnis eine bestimmte Wahrscheinlichkeit an, dem Programmierer ist dann überlassen, ab welchem prozentualen Wert ein Ergebnis als richtig definiert werden soll. In der Regel wird die höchste Prozentzahl als richtig gewählt, dies kann aber häufig falsch sein, wenn diese Lösung z.B. nur mit sehr knappem Vorsprung vor einer anderen Lösung steht. Dieses Problem besteht bei PSO zwar grundsätzlich auch, durch die Fitnessfunktionen kann dieser Abstand jedoch deutlich größer gemacht werden. Es können also systematisch bestimmte Suchergebnisse auf- oder abgewertet werden. Dies ist bei HM-Modellen nicht möglich, es können lediglich Anfangsgewichtungen vergeben werden, die das Modell während des Trainingsprozesses selbst anpasst. So bleibt bei einer hohen Fehlerquote keine Möglichkeit, korrigierend einzugreifen.

Ein weiterer Vorteil des Partikelschwarmalgorithmus liegt in der zukünftigen Entwicklung der Rechenkraft im Computerbereich. Einem HM-Modell wird diese Entwicklung kaum weiteren Raum bieten, da ja der Trainingsprozess vor dem eigentlichen Tracking abläuft und deshalb nicht als zeitkritisch einzustufen ist. Der PSO-Algorithmus kann dagegen von jeder Leistungssteigerung profitieren, da das Training hier während des Suchprozesses und damit unter Echtzeitanforderungen zu bewältigen ist. Die Fitnessfunktionen könnten dann komplexer werden und weitere Kriterien hinzunehmen bzw. es könnten bei einem adaptiven Verfahren generell mehr Fitnessfunktionen eingesetzt werden. Hier wäre es dann möglich, auf bestimmte Situationen und Abfolgen, die sonst schwer nachzuführen sind, mit mehreren, an die Aufgabe angepassten Fitnessfunktionen heranzugehen.

Es bleibt also festzuhalten, dass der Partikelschwarmalgorithmus auch mit den in dieser Arbeit vorgestellten Neuerungen beste Voraussetzungen hat, das wichtigste Mit-

tel im Bereich Bildverarbeitung und Videonachführung zu werden. Gegenüber Kantendetektion oder wieder bereits abflauenden Ideen wie dem Optischen Fluss, der einst als große Zukunftshoffnung gesehen wurde, ist jedes PSO-basierende System bereits heute im Vorteil. Das aktuell wichtigste Instrument, das HM-Modell, wird in Zukunft wohl deutlich weniger neue Impulse erhalten, da die grundlegende Idee als abgeschlossen gilt und die Veränderungen hauptsächlich nur an Dimensionierungen stattfinden, z.B. der Gewichtung bestimmter Zustände oder der Anzahl der Verbindungen. Der Bereich PSO wird dagegen vor allem in den letzten Jahren intensiv erforscht. Neben den hier vorgestellten Ideen gibt es zahlreiche weitere wie hierarchische Schwärme oder Verbesserungen an den Grundlagenformeln des PSO, die versuchen, die Idee selbst zu verbessern. Vor allem in Situationen, die eine hohe Komplexität aufweisen, sind dabei Verbesserungen entstanden.

Neben den beschriebenen Nachführungsaufgaben kann der PSO in einer Reihe weiterer Anwendungsmöglichkeiten führend werden: Etwa die Optimierung mathematischer Funktionen stellt ein wichtiges Feld dar, vor allem in Situationen, in denen andere Verfahren nicht schnell genug oder nicht genau genug zum Ziel führen können. Eventuell wird man dem Konzept PSO in Zukunft weitere Anwendungsgebiete erschließen. Exemplarisch wäre es vorstellbar, aus einer großen Anzahl Frames bestimmte Frames herauszusuchen, die bestimmte Informationen enthalten. So könnte etwa gespeichertes Videomaterial in Sicherheitsanwendungen zielsicher ausgewertet werden.

Bibliography

[1] J. Kennedy and R. C. Eberhart. Particle swarm optimization. *Proc. IEEE International Conference Neural Networks*, 4:1942–1948, 1995.

[2] Xiaohui Hu. Particle Swarm Optimization: Tutorial. swarmintelligence.org.

[3] Archetypen/Urfiguren in allen Kulturkreisen. Wikipedia.

[4] Ameisenalgorithmus. Wikipedia.

[5] R.C. Eberhart J. Kennedy and Y. Shi. *Swarm Intelligence*. Academic Press, 2001.

[6] A. Carlisle and G. Dozier. An off-the-shelf pso. Proc. Workshop on Particle Swarm Optimization, Indianapolis, 2001.

[7] I. C. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317–325, 2003.

[8] Yuhua Zheng and Yan Meng. The pso-based adaptive window for people tracking. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)*, 2007.

[9] PSO Applications. Wikipedia.

[10] J. Canny. A computational approach for edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[11] Florian Obermeier Tobias Oelbaum. Bewegungssuche mittels blockmatching. Vorlesung digitales Video, TU München, 2004/2005.

[12] M. Lang G. Rigoll. Kurzmanuskript mensch-maschine-kommunikation 2. Vorlesung Mensch-Maschine-Kommunikation 2, TU München, 2004.

[13] David A. Forsyth Deva Ramanan and Andrew Zisserman. Tracking people by learning their appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):65–81, 2007.

[14] Brian G. Schunck Bertold K. P. Horn. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.

[15] M.Sarkis K. Diepold, F. Keyrouz. Skript computer vision. Vorlesung Computer Vision, Lehrstuhl für Datenverarbeitung, TU München, 2005.

[16] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[17] G. Welch and G. Bishop. An introduction to kalman filter. ACM SIGGRAPH 2001 Course #8, 2001.

[18] Brian Birge. Particle swarm optimization toolbox. http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=7506, 2005.

[19] Matthias Scheutz. Real-time hierarchical swarms for rapid adaptive multi-level pattern detection and tracking. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)*, 2007.

[20] Jim Pugh and Alcherio Martinoli. Inspiring and modeling mulit-robot search with particle swarm optimization. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)*, 2007.