Diplomarbeit

Universität Bielefeld
Technische Fakultät

Online Klassifikation menschlicher Aktionen anhand von Space-Time Interest-Points

Abschlußarbeit zur Erlangung des akademischen Grades
Diplom-Informatiker

Maximilian Panzner
April 2013

Referent: Prof. Dr. Philipp Cimiano
Korreferent: Dipl. Inf. Oliver Beyer

Abstract

Human motion classification is an important area of computer vision with a variety of applications in surveillance, human-computer interfaces and robotics. Many current systems for human motion classification rely on a batch processing scheme to learn their classification model. This excludes these systems from many possible applications where fast response to new classes of stimuli is necessary. This thesis will present two approaches for incremental online classification of human motion, which will allow the system to adapt to new situations on the fly, without the need to go through the whole batch learning process again. The developed algorithms are tested against a current state of the art offline classification system, which already has shown good results on several human motion databases. It will be shown, that the developed online classification systems can archive competitive results while avoiding several limitations of the offline approaches.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

In der folgenden Arbeit wird ein System zur *online* Klassifikation von menschlichen Aktionen entwickelt und evaluiert. Das System extrahiert zunächst aktionsrelevante Merkmale aus eingehenden Videobildern, wobei nicht nur räumliche Eigenschaften der Bilddaten berücksichtigt werden, sondern auch deren zeitlicher Verlauf. Sequenzen dieser Merkmale können verschiedenen zuvor gelernten Aktionsklassen zugeordnet werden. Das Verfahren ist dabei so angelegt, dass jederzeit neue Trainingsdaten integriert werden können, ohne sie explizit speichern zu müssen. Es können auch bisher vom System noch nicht gesehene Aktionsklassen im laufenden Betrieb gelernt werden.

1.1. Motivation

Heutzutage fallen in vielen Bereichen immer größere Mengen an Videodaten an. Dazu gehören beispielsweise Videoportale wie Youtube oder MyVideo. Allein Youtube zählt nach eigenen Angaben schon über 4 Milliarden aufgerufene Videos pro Tag.[1] Auch in sozialen Netzwerken wie Facebook und Google+ werden von den Nutzern immer mehr Videos geteilt. Zu diesen Videos sind meist nur Metadaten wie der Zeitpunkt der Aufnahme, die Länge der Aufnahme oder die GPS Koordinaten bekannt. Im besten Fall wurden die Videos von den Nutzern noch mit Schlagworten versehen, die die Aufnahmen in wenigen Worten kurz beschreiben. Um diese Bestände sinnvoll und durchsuchbar zu strukturieren, ist es von Vorteil, nicht nur die Metadaten des Videos zu kennen, sondern auch zu wissen, was in den einzelnen Videos passiert. Das Wissen darüber, welche Aktionen von eventuell abgebildeten Personen ausgeführt werden, kann einen bedeutenden Beitrag zur semantischen Einordnung des entsprechenden Videos leisten.

[1] http://www.reuters.com/article/2012/01/23/us-google-youtube-idUSTRE80M0TS20120123[Reu].

Ein weiteres Anwendungsfeld der Aktionsklassifikation ist die Videoüberwachung. Ein automatisches System, das nicht nur Bewegung an sich meldet, sondern auf bestimmte verdächtige Bewegungsmuster reagiert, könnte die Auswertung von Überwachungsvideos bedeutend vereinfachen. So müsste ein automatisches Überwachungssystem zum Beispiel Personen erkennen können, die ihr Gepäck zurücklassen oder es in einen Mülleimer stecken. Solche Verhaltensweisen könnten den Verdacht eines Sprengstoffanschlages begründen.

Auch in der Robotik gibt es zahlreiche Einsatzmöglichkeiten für semantische Videoklassifikation. Vor dem Hintergrund der ständig alternden Gesellschaften zeichnet sich ab, dass in absehbarer Zeit auch technische Systeme in der Altenpflege eingesetzt werden und dem Pflegepersonal ermöglichen, sich auf die zwischenmenschlichen Aspekte zu konzentrieren. In der unten stehenden Abbildung ist ein Serviceroboter zu sehen, der in diesem Umfeld eingesetzt werden könnte.

Der SCITOS-G3 von Metralabs ist ein Roboter, der z.B. zur Unterstützung leicht dementer älterer Menschen gedacht ist. Das System soll diesen Menschen ermöglichen, länger zu Hause in ihrer gewohnten Umgebung leben zu können. Als Demonstrationsplattform wird der Roboter bisher als Telepräsenzsystem eingesetzt, über das jederzeit ein Arzt zugeschaltet werden kann. Des Weiteren erinnert das System an die Einnahme von Medikamenten und fordert die Patienten regelmäßig zu kleinen kognitiven Trainingsspielen auf, um die Entwicklung der Demenz zu verzögern. Bei diesem Einsatzgebiet wäre es durchaus denkbar, den Roboter mit einem Aktionsklassifikationssystem auszustatten, mit dem der Roboter beispielsweise erkennen kann, wenn der Patient sich aus einer Zwangslage nicht mehr selbst befreien kann.

1.1.1. Klassifikation menschlicher Aktionen

Klassifikation menschlicher Aktionen ist ein Prozess, bei dem zunächst ein Modell der charakteristischen Eigenschaften der verschiedenen Aktionen gelernt werden soll. Der Klassifikator soll mit diesem Modell letztendlich in der Lage sein, die in einer Videosequenz vorkommenden Aktionen unterscheiden und benennen zu können. So soll der Klassifikator z.B. alle Video-Anfragen, in denen Personen zu sehen sind, die sich auf ihren Füßen fortbewegen, mit „gehen“ beantworten. Sequenzen, auf denen sich Personen mit Inline Skates fortbewegen, sollen hingegen mit „skaten“ beantwortet werden.

Aktionssequenzen zu klassifizieren ist eine komplexe Aufgabe. Die selben Aktionen können von verschiedenen Akteuren sehr unterschiedlich ausgeführt werden. Weitere Schwierigkeiten entstehen durch Variationen in den Aufnahmebedingungen. So können sich die Lichtverhältnisse abrupt ändern, wenn sich z.B. eine Wolke vor die Sonne schiebt. Handelnde

Personen können ihre Entfernung zur Kamera ändern. Eigenbewegungen der Kamera können schwer zu lösenden Mehrdeutigkeiten hervorbringen. So kann ein Objekt z.B. statisch erscheinen wenn sich die Kamera mit gleicher Geschwindigkeit in die selbe Richtung bewegt. Bei sehr dynamischen Hintergründen kann es schon herausfordernd sein, handelnde Personen überhaupt von sich bewegenden Hintergrundstrukturen zu unterscheiden.

1.1.2. Einschränkungen gängiger Verfahren

Viele der gängigen Verfahren zur Klassifikation menschlicher Aktionen setzen darauf, die gemeinsamen Eigenschaften der zu klassifizierenden Aktionen in einem separaten Trainingsprozess zu lernen. Neues Wissen lässt sich so im laufenden Betrieb nicht integrieren. Dies bringt einige Einschränkungen mit sich:

- Ändert sich das Einsatzumfeld des Systems, muss es gegebenenfalls außer Dienst gestellt und komplett neu trainiert werden (*batch learning*). Hierfür müssen alle bisherigen Trainingsbeispiele explizit gespeichert werden.
- Ist der Trainingsdatensatz so umfangreich, dass er nicht mehr in den schnellen Hauptspeicher passt, steigt der Laufzeitbedarf vieler iterativ arbeitenden Verfahren stark an. Dies liegt unter anderem darin begründet, dass die Trainingsdaten in jeder Iteration wiederholt aus einem langsamen Massenspeicher nachgeladen werden müssen.

1.2. Ziel der Arbeit

Im Rahmen dieser Arbeit soll zunächst ein bereits existierendes *offline* System zur Klassifikation menschlicher Aktionen implementiert werden, das bereits gute Ergebnisse in verschiedenen Szenarien gezeigt hat. Dieses System soll zu einem *online* Klassifikationssystem erweitert werden, dass die im letzten Abschnitt genannten Einschränkungen nicht aufweist. Dabei sollen folgende zentrale Anforderungen umgesetzt werden:

- **Inkrementelles Lernen.** Das System soll inkrementell jeweils ein eingehendes Datum nach dem anderen verarbeiten.
- **Adaptives Lernen**. Das System soll sich moderaten Änderungen in der Aktionsausführung mit der Zeit anpassen können.
- **Erweiterbarkeit**. Das System soll jederzeit im laufenden Betrieb durch neue Trainingsbeispiele erweitert werden können.

Durch das inkrementelle Lernen entfällt die Notwendigkeit Trainingsbeispiele explizit speichern zu müssen. Das System betrachtet immer einen Datenpunkt nach dem nächsten, eine globale Sicht auf die Daten ist nicht erforderlich. Das System kann somit unabhängig von der Menge des verfügbaren Hauptspeichers skalieren. Durch die Forderung nach dem adaptiven Lernen soll dem System ermöglicht werden, sich moderaten Änderungen in der Ausführung bereits gelernter Aktionen anzupassen. Darüber hinaus soll das System auch bisher nicht beobachtete Aktionsklassen lernen können ohne dafür außer Dienst gestellt werden zu müssen.

In dieser Arbeit werden zwei Ansätze entworfen und implementiert, die diese Forderungen umsetzen.

1.3. Gliederung

Diese Arbeit gliedert sich in insgesamt 10 Kapitel. In Kapitel 2 werden die Grundlagen und die allgemeine Vorgehensweise der Aktionsklassifikation dargelegt. Daran schließt sich in Kapitel 3 die Vorstellung eines bereits existierenden Verfahrens zu *offline* Aktionsklassifikation von Laptev et al. an. Im Hauptteil der Arbeit wird in Kapitel 4 zunächst das zuvor vorgestellte *offline* Verfahren implementiert. Darauf folgt in Kapitel 5 die Evaluation des Verfahrens und der Vergleich der Eigenimplementation mit der Referenzimplementation von Laptev. Kapitel 6 analysiert zunächst die Einschränkungen des *offline* Klassifikationssystems und legt die Anforderungen an die zu entwickelnden Ansätze zur *online* Klassifikation dar. In den Kapiteln 7 und 8 werden zwei Vorschläge zur Umsetzung eines *online* Klassifikationssystems gemacht, die in Kapitel 9 evaluiert und diskutiert werden. Das letzte Kapitel 10 schließt die Arbeit mit einem Fazit ab und gibt einen Ausblick zur möglichen Weiterentwicklung des Systems.

1.4. Lesehinweis

Nach Möglichkeit verwendet der folgende Text deutsche Begriffe, manche Fachwörter haben im Deutschen jedoch keine adäquate Entsprechung. Diese Fachbegriffe werden in englischer Sprache *kursiv* gesetzt und im Text kurz eingeführt, sofern es sich nicht um allgemeingültige Begriffe in der Informatik oder speziell der Bildverarbeitung handelt. Die Adressen von Webseiten wurden im April 2013 ermittelt und letztmalig geprüft. Für eine längerfristige Verfügbarkeit der Seiten kann nicht garantiert werden.

Kapitel 2

Grundlagen der Aktionsklassifikation

Aktionsklassifikation, wie sie hier betrachtet werden soll, ist der Prozess aus zeitlichen Bildsequenzen $f : \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}$ mit den beiden räumlichen Dimensionen x, y und der zeitlichen Dimension t auf die darin abgebildeten Aktionen zu schließen. Obwohl in den letzten Jahren einige Ansätze zur Klassifikation menschlicher Aktionen publiziert worden sind, ist es immer noch ein aktives Feld der aktuellen Forschung. Nicht zuletzt auch aufgrund der vielfältigen Herausforderungen, die sich bei der Aktionsklassifikation zeigen.

Ein dynamischer Hintergrund macht es z.B. schwierig, die eigentlich relevanten Bewegungen zu finden und zur weiteren Verarbeitung zu isolieren. Änderungen der Beleuchtungssituation können Phantombewegungen induzieren, da sich bewegende Schatten nur schlecht von tatsächlichen Objektbewegungen unterscheiden lassen. Eigenbewegungen der Kamera können schwer zu lösende Mehrdeutigkeiten in den beobachteten Bewegungsmustern begünstigen. Ein Objekt kann z.B. statisch erscheinen, wenn sich die Kamera mit gleicher Geschwindigkeit in die selbe Richtung bewegt. Unterschiedliche Akteure führen die selben Aktionen mitunter völlig verschieden aus, was sich in hoher Intra-Klassen Varianz zeigt. Vordergrundobjekte können die Aktionen verdecken, so dass sie nicht in ihrer ganzen räumlichen und zeitlichen Ausdehnung beobachtet werden können.

Aktionen

Um Aktionen klassifizieren zu können, ist es zunächst einmal sinnvoll, genau einzugrenzen, was eine Aktion ist. Die hier verwendete Hierarchie menschlicher Bewegungen lehnt sich an Bobick et. al[BDSS01] und Moeslund et. al[MHK06] an.

> **Aktionsprimitiv** ist eine atomare Bewegung, deren Ausführung einfach ist und sich leicht durch ihre Raum-Zeit Trajektorie beschreiben lässt; Z.B. einen Fuß auf den Boden stellen.

Aktion ist eine Sequenz aus atomaren Bewegungen; Z.B. „laufen“ als zyklische Sequenz von „anheben“, „vorsetzen“ und „absenken“ Bewegungen des Beines.

Aktivität beschreibt komplexe Interaktionen von mehreren Aktionen. Aktivitäten können auch mehrere handelnde Personen umfassen; Z.B. „Volleyball spielen“.

Verfahren

Die rohen Pixeldaten der zu untersuchenden Videosequenz sind nicht sehr informativ im Hinblick auf die Klassifikation menschlicher Aktionen. Sie sind auch nicht besonders stabil gegenüber Umwelteinflüssen. Schon die Änderung der Beleuchtung führt zu massiven Änderungen in der Aufnahme. Die Videodaten sollten zunächst so vorverarbeitet werden, dass nicht relevante Informationen weitgehend eliminiert werden und den weiteren Prozess der Klassifikation nicht stören. Praktisch wird dies meist so umgesetzt, dass auf den Videodaten an bestimmten Stellen sogenannte Deskriptoren berechnet werden, die die für die Klassifikation relevanten Informationen wesentlich dichter kodieren als die Rohdaten. Ganz allgemein lässt sich die Klassifikation menschlicher Aktionen in die folgenden drei Phasen einteilen:

- **Detektion** von interessanten Regionen oder Primitiven im Videobild.
- **Repräsentation** der rohen Pixeldaten durch aussagekräftige Deskriptoren.
- **Klassifikation** dieser Deskriptoren in die verschiedenen Aktionsklassen.

Diesem Schema folgend, werden zuerst aus der vorliegenden Videosequenz möglichst aussagekräftige Informationen über die Bewegungen der handelnden Personen extrahiert. Diese Informationen repräsentieren dann die zu untersuchende Videosequenz, anstatt der rohen Pixeldaten. Im letzten Schritt werden diese Informationen interpretiert, so dass am Ende ein eindeutiges Aktionslabel steht.

2.1. Detektion von Aktionsprimitiven

Wie zuvor erwähnt, sollen die rohen Pixeldaten der Videosequenz zu einer Sequenz möglichst aussagekräftiger Deskriptoren verdichtet werden. Zur Auswahl der Stellen im Video, an denen diese Deskriptoren berechnet werden sollen, gibt es zwei Vorgehensweisen. Die einfachste Möglichkeit ist es, auf einen aufwändigen Detektionsschritt zu verzichten und die Videosequenz in festen Abständen oder an einem regulären Gitter abzutasten, das sogenannte *dense sampling*. Eine andere Möglichkeit ist es zu versuchen, markante Punkte im Bild zu finden. Diese Punkte liegen z.B. an Stellen, die besonders viel über die gesuchte Struktur im Bild verraten oder die besonders stabil gegenüber störenden Einflüssen wie unterschiedlichen Aufnahmewinkeln oder Beleuchtungsänderungen sind. Diese Punkte

werden *interest points* genannt und sollen im folgenden Abschnitt genauer definiert werden. Einen umfangreichen Vergleich zwischen *dense sampling* und *sparse sampling* mittels *interest points* bietet Wang et. al [WUK+09]. Allgemein lässt sich sagen, dass beim *dense sampling* mehr Kontextinformation aus dem Hintergrund der Bewegung extrahiert wird, wodurch dem Klassifikationsprozess letztendlich die Aufgabe zukommt, Wichtiges von Unwichtigem zu trennen. Die Detektion von *interest points* lässt sich als solche schon als Auswahlprozess sehen und führt der Klassifikation damit eine möglichst relevante Vorauswahl an Beispielen zu.

2.1.1. Interest Points

Ein *interest point* ist ein Punkt mit einer wohldefinierten Position im Bildraum, der sich robust und reproduzierbar detektieren lässt. Das bedeutet insbesondere, dass die Detektion der *interest points* stabil gegenüber kleinen Variationen und Störungen im Bild oder geändertem Aufnahmewinkel sein sollte. Das kann z.B. ein lokales Maximum der Helligkeit des Bildes oder der Schnittpunkt zweier Linien sein. Die Kriterien für einen *interest point* sollten so gestaltet werden, dass die erkannten Punkte an für die Klassifikationsaufgabe möglichst aussagekräftigen Stellen liegen. Zur Klassifikation von Objekten in 2D Bildern haben sich z.B. Ecken als informativ und stabil erwiesen.

2.2. Repräsentation von Aktionsprimitiven

Nachdem die für die Klassifikationsaufgabe interessanten Stellen gefunden sind, müssen sie geeignet repräsentiert werden. Meist wird dabei nicht nur der detektierte Punkt an sich, sondern die ganze Nachbarschaft um diesen Punkt herum betrachtet.

Da die berechneten Deskriptoren im Allgemeinen deutlich niedrigdimensionaler sind als der Vektor der rohen Pixeldaten, kann man Repräsentationsverfahren auch als eine Art von Dimensionsreduktion auffassen. Diese Verfahren lassen sich grob in die zwei Kategorien globale Repräsentation und lokale Repräsentation unterteilen. Dieser Abschnitt soll einen Überblick über die geläufigsten Verfahren zur Bildrepräsentation geben.

2.2.1. Globale Bildrepräsentation

Globale Bildrepräsentation betrachtet das Videobild als Ganzes. So kann beispielsweise eine Person durch Hintergrundsubtraktion und *tracking* lokalisiert werden. Der Umriss

der Person bildet dann eine *region of interest*[2], innerhalb derer mit modellbasierten Verfahren nach der Silhouette der Person gesucht werden kann. Ist die Silhouette gefunden, kann man die Pose der Person schätzen und zu einem numerischen Wert, dem Deskriptor, verrechnen.

Space-Time Volumes

Space-time volumes sind 3 dimensionale Volumina, die sich ergeben, wenn man die zweidimensionalen Einzelbilder einer Videosequenz hintereinander legt. Die Basis der Verfahren, die mit *space-time* Volumen arbeiten, ist es Deskriptoren auf diesen Volumen zu berechnen und deren Ähnlichkeit zueinander zu betrachten. Haben zwei Volumen, bzw. die auf ihnen berechneten Deskriptoren, eine gewisse Ähnlichkeit, kann man annehmen, dass auch die abgebildeten Bewegungen ähnlich sind. Manche Verfahren arbeiten direkt mit den rohen Pixeldaten, andere extrahieren zunächst bestimmte Bildstrukturen. Bobick und Davis[BDSS01] verwenden z.B. Silhouetten und aggregieren deren zeitliche Änderungen zu einem *motion history image* (MHI), das die Bewegungshistorie als *recency* Funktion der Bewegung kodiert. Die entstehenden MHIs dienen dann als *template* und werden zur Klassifikation mit bereits bekannten *templates* abgeglichen. Da das *template matching* (Abschnitt 2.3) mit vielen zu unterscheidenden Aktionen sehr aufwändig ist, wird zunächst eine Vorauswahl an Aktionen zusammengestellt, die in der aktuellen Sequenz mit einer gewissen Wahrscheinlichkeit vorkommen. Dazu wird ein binäres *motion energy image* (MEI) berechnet, das an Stellen, an denen Bewegung stattfindet eine 1 aufweist. Das MEI lässt sich durch Differenzbildung und Anwendung eines geeigneten Schwellwerts leicht berechnen. Letztendlich lassen sich mit dem *motion energy image*, als vereinfachte Repräsentation, schnell Vorschläge für mögliche Bewegungen in der aktuellen Videosequenz generieren.

2.2.2. Lokale Bildrepräsentation

Im Gegensatz zu den globalen Repräsentationen nutzen die lokalen nur Informationen aus der direkten Nachbarschaft der Position, an der sie erfasst werden. Es müssen also keine globalen Strukturen wie Personen oder Objekte gesucht werden. Das umgeht z.B. zahlreiche Probleme der Hintergrundsubtraktion oder der genauen Lokalisierung und Abgrenzung dieser Strukturen. Nach einer Analyse von R. Poppe [Pop10] sind lokale Repräsentationsverfahren dadurch etwas widerstandsfähiger gegenüber Änderungen des Kamerawinkels, dem Aussehen der Person und teilweiser Überdeckung durch Vordergrundobjekte. Die in dieser Arbeit verwendeten lokalen Repräsentationsverfahren werden in Kapitel 3 eingehend vorgestellt.

[2] *region of interest*, kurz *ROI* ist ein Ausschnitt aus dem Bild, der für die aktuelle Aufgabe interessante Informationen enthält.

2.3. Klassifikation von Aktionsrepräsentationen

Die Klassifikation ist der letzte Schritt bei der Erkennung von Aktionen. Hierbei sollen die extrahierten Repräsentationen mit vorher gelernten Beispielen oder daraus gewonnenen Modellen verglichen werden. Durch diesen Vergleich wird entschieden, welcher Aktionsklasse das Eingabebeispiel zugeordnet werden soll.

Template Matching

Eine einfache Form der Aktionsklassifikation ist das *template matching*. Hierbei werden zu klassifizierende Aktionsrepräsentationen mit bereits bekannten Mustern aus der Trainingsphase verglichen. Dem eingehenden Muster wird das Aktionslabel des ähnlichsten Trainingsbeispiels zugewiesen. Die Schwierigkeit bei diesen Verfahren liegt meist darin, ein geeignetes Ähnlichkeitsmaß zu finden. Bei vektoriellen *features* kann das z.B. der euklidsche Abstand sein. Bei sequenziellen *features* wird der Vergleich schon aufwändiger. Als Beispiel sollen zwei Sportler einen Karatetritt ausführen. Beide Aktionsausführungen werden mit einer Kamera mit 25 Bildern pro Sekunde aufgenommen. Aus den Einzelbildern wird die Pose der Sportler ermittelt und als Vektor der Gelenkwinkel repräsentiert. Der erste Sportler führt die Bewegung schnell aus und ist nach einer Sekunde fertig. Bei 25 Bildern pro Sekunde liegt also eine Sequenz aus 25 Messungen der Gelenkwinkel vor. Der zweite Sportler führt die Bewegung nun langsam aus und braucht zwei Sekunden. Insgesamt ist die zweite Sequenz also durch 50 Einzelmessungen repräsentiert. Die beiden Sequenzen können auf Grund ihrer unterschiedlichen Länge nicht direkt miteinander verglichen werden. Hier kommen Algorithmen wie z.B. das *dynamic time warping* zum Einsatz, die Sequenzen unterschiedlicher Länge miteinander vergleichen können.

Diskriminative Klassifikatoren

Diskriminative Klassifikationen lernen Trennfunktionen, die verschiedene Klassen voneinander unterscheiden können. Dieser Ansatz bietet bessere Generalisierungseigenschaften als das einfache *template matching* und kommt ohne ein explizites Modell, wie z.B. die Pose eines Menschen, aus. Ein Verfahren aus dieser Klasse ist die *support vector machine*, die in Abschnitt 3.4.1 eingehend vorgestellt wird.

Kapitel 3

Offline Aktionsklassifikation nach Laptev

Dieses Kapitel stellt eine Klassifikationspipeline von Ivan Laptev und Kollegen vor, die bereits vielversprechende Ergebnisse sowohl in synthetischen Testszenarien als auch in komplexen Ausschnitten aus Hollywoodfilmen gezeigt hat [Lap05, LMSR08, Lap04].

Das vorgestellte Verfahren beinhaltet Lösungen für alle drei Phasen der Aktionsklassifikation:

- **Detektion**
  Zuerst werden für die Aktionsklassifikation interessante Punkte als *space-time interest points* im Video detektiert. Diese Form der *interest points* berücksichtigen zusätzlich zu den räumlichen Eigenschaften der Bildsequenz auch deren zeitlichen Verlauf.

- **Repräsentation**
  Auf den Nachbarschaften dieser Punkte werden aus den Pixeldaten Deskriptoren berechnet. Diese Deskriptoren kodieren sowohl Informationen über die Texturierung der Umgebung um diesen Punkt als auch Informationen über Bewegungen von Bildstrukturen, die in dessen Umgebung aufgetreten sind. Die berechneten Deskriptoren werden anschließend in einem Clustering-Schritt nach Ähnlichkeit zusammengefasst und durch die Nummer ihres Clusters ersetzt.

- **Klassifikation**
  Die Clusternummern werden in einem normalisierten Histogramm aufgetragen und letztendlich mittels einer SVM klassifiziert.

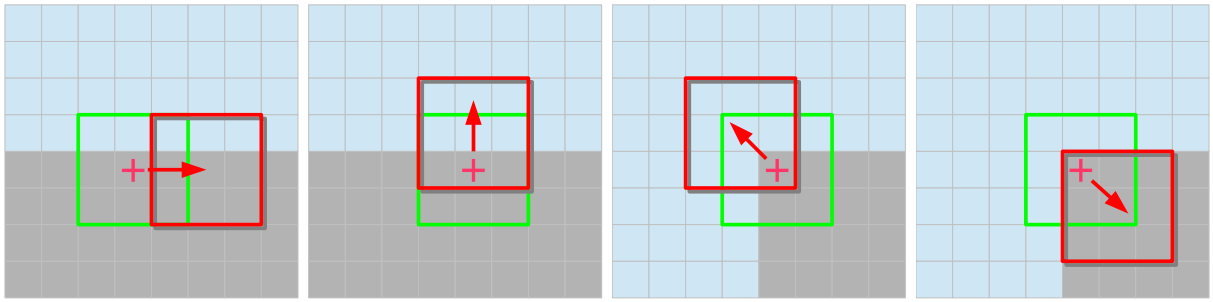


Abbildung 3.1.: Diskrete Verschiebung der Fensterfunktion w um den aktuellen Pixel. Von links nach rechts: 1) entlang einer Kante, 2) orthogonal zu einer Kante, 3) von einer Ecke weg, 4) zu einer Ecke hin.

3.1. Entwicklung des Interest Point Detektors

Um die zentralen Ideen des in dieser Arbeit genutzten *space time interest point* Detektors zu erläutern, ist es hilfreich, sich die Historie dieses Detektors anzusehen. Wie in Abschnitt 2.1.1 eingeführt, zeichnen sich *interest points* durch eine solide mathematische Definition aus, die eine hohe Reproduzierbarkeit der Detektion auch unter leicht abgewandelten Aufnahmebedingungen ermöglichen soll. Eine beliebte zweidimensionale Form der *interest points* sind Ecken. Sie lassen sich als gemeinsamer Endpunkt zweier Kanten charakterisieren. Im Folgenden sollen zwei aufeinander aufbauende Detektoren zur Erkennung von Ecken in statischen zweidimensionalen Bildern vorgestellt werden.

3.1.1. Moravec Corner-Detector

Moravecs Detektor[Mor80] fasst die zu findenden Ecken als Punkte im Bild auf, die eine geringe Ähnlichkeit zu sich selbst aufweisen. Hohe Selbstähnlichkeit ist z.B. gegeben, wenn die Umgebung des Punktes in alle Richtungen sehr homogen ist. Um diese Selbstähnlichkeit zu berechnen, legt der Detektor um jeden Punkt des Bildes f ein lokales Fenster w und berücksichtigt die Änderungen der Bildintensität, die entstehen, wenn man dieses Fenster um einen kleinen Versatz (u, v) in verschiedene Richtungen verschiebt. Das Maß für die Selbstähnlichkeit berechnet sich dabei als quadratische Differenz der Pixelwerte unter diesen beiden Fensterpositionen wie folgt:

$$E_{x,y} = \sum_{u,v} w_{u,v}(f_{x+u,y+v} - f_{x,y})^2 \tag{3.1}$$

Dabei gibt es drei Fälle zu unterscheiden:

- Die Bildstruktur innerhalb des Fensters ist flach. Kleine Verschiebungen des Fensters ergeben nur kleine Änderungen der Bildintensität.
- Das Fenster liegt über einer Kante. Verschiebungen entlang der Kante ergeben nur kleine Änderungen. Verschiebungen orthogonal zur Kante ergeben jedoch große Änderungen (Abbildung 3.1, Bild 1 und 2).
- Das Fenster liegt über einer Ecke. Verschiebungen ergeben große Intensitätsänderungen in alle Richtungen (Abbildung 3.1, Bild 3 und 4).

Die Operatorantwort ist das Minimum der Intensitätsänderungen aller Richtungen. Eine Ecke liegt vor, wenn die Operatorantwort über einem Schwellwert liegt und zugleich ein Maximum in seiner lokalen Umgebung einnimmt.

3.1.2. Harris Corner-Detector

Der Harris Corner-Detector ist eine Weiterentwicklung des Moravec Corner-Detectors. Harris und Stephens [HS88] sahen drei Probleme mit Moravecs Operator:

- der Operator neigt zu Rauschen, da die verwendete Fensterfunktion rechteckig ist und alle Pixel gleich gewichtet werden.
- die Operatorantwort ist anisotrop, da die Verschiebungen nur in diskreten Richtungen (z.B. alle 45°) berechnet werden.
- Da der Operator jeweils nur das Minimum der Antworten in den verschiedenen Richtungen betrachtet, wird die Rauschneigung zusätzlich verstärkt.

Um die Operatorantwort zu verbessern und die Rauschneigung zu vermindern, schlagen Harris und Stephens vor, die rechteckige Fensterfunktion durch eine kreisförmige Gaußfunktion zu ersetzen. Damit die Antwort möglichst isotrop ausfällt, berechnen Harris und Stephens die Verschiebungen nicht mehr nur in diskrete Richtungen, sondern gehen einen analytischen Weg.

Momentenmatrix

Zuerst werden die Gradienten L_x und L_y z.B. mit dem Sobel Operator approximiert. Die Änderungen der Bildintensität $E_{x,y}$ (Formel 3.1) für kleine Verschiebungen (u, v) können auch als

$$E_{x,y} = (u, v)M(u, v)^T \tag{3.2}$$

geschrieben werden. Wobei die symmetrische 2×2 Matrix

$$M = \begin{pmatrix} L_x^2 & L_x L_y \\ L_y L_x & L_y^2 \end{pmatrix}$$

die Autokorrelationsmatrix der Gradientenrichtungen in der Umgebung des Punktes (x, y) ist. Diese Matrix entspricht der Momentenmatrix zweiter Ordnung und wird häufiger auch Strukturtensor genannt. Die Eigenwerte λ_1 und λ_2 von M beschreiben nun die Variation der Bildintensitäten in f entlang und senkrecht zur Gradientenhauptrichtung.
Sind λ_1 und λ_2 beide signifikant groß, so liegt in der Bildfunktion an der Stelle (x, y) ein Interest-Point. Da die Eigenwerte nicht direkt benötigt werden, sondern nur ihr Verhältnis, schlagen Harris und Stephens vor, die *interest points* als positive Maxima der Operatorfunktion

$$H = det(M) - k\ trace^2(M) = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{3.3}$$

zu detektieren. In der Praxis hat sich der Faktor $k = 0.04$ bewährt.

3.2. Detektion von Space-Time Interest Points

Space-Time Interest Points erweitern das Konzept der *interest points* auf Bewegtbilder. Basis der in dieser Arbeit entwickelten Verfahren ist der Harris3D Detector, wie er von Ivan Laptev in seinem Paper „on space-time interest-points“ [Lap05] vorgeschlagen wurde. Der Harris3D Detektor arbeitet auf Grauwert-Bildsequenzen und reagiert auf ganz bestimmte Punkte (x, y, t) in Bewegtbildern, die ein Maximum an Variabilität in ihrer lokalen Umgebung in Raum und Zeit aufweisen. Im 2D-Fall des Harris Corner Detektors waren diese besonders variablen Bildpunkte vor allem an Ecken zu finden. Wenn als dritte Dimension in Bewegtbildern die Zeit hinzu kommt, finden sich diese „Ecken“ insbesondere an Punkten, deren Bildstrukturen nicht konstanter Bewegung unterliegen.

Um dies an einem konkreten Beispiel zu verdeutlichen, kann man als Bildstruktur die Hände der klatschenden Person aus Abbildung 3.2(b) betrachten. Die Punkte mit der höchsten Variabilität sind zu erwarten, wenn sich die beiden Hände in der Mitte treffen und sich die Bewegung zum erneuten Klatschen umkehrt. Die räumliche Variabilität ergibt sich dabei durch die Texturierung der Hand im Gegensatz zu einer homogen gefärbten Fläche. Die Variabilität in zeitlicher Richtung ist durch die Umkehr der Bewegungsrichtung gegeben. In dem in Abbildung 3.2(b) unten gezeigten Schnitt durch das Raum-Zeit-Volumen der kurzen Sequenz sind die erkannten Punkte als Schwerpunkte der eingezeichneten Ellipsen

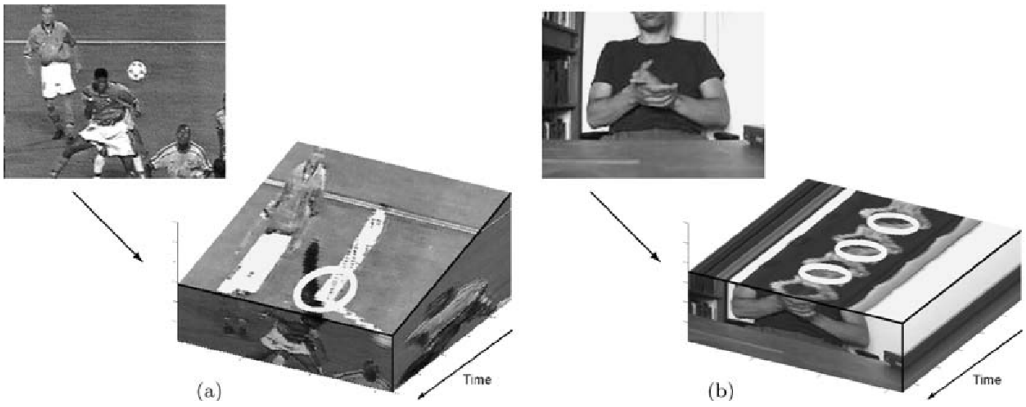
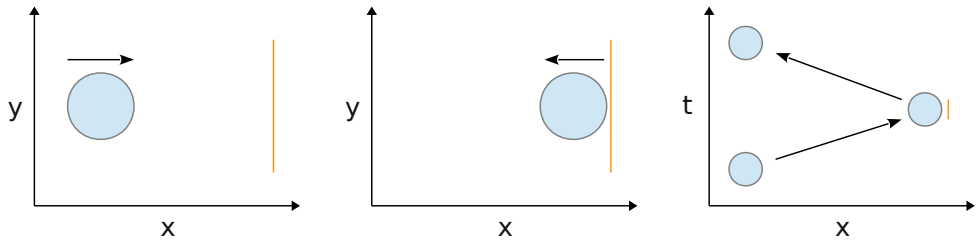


Abbildung 3.2.: Schnitte durch die Raum-Zeit Volumina zweier Bildsequenzen. Die abgebildeten Raum-Zeit Volumina ergeben sich durch Hintereinanderlegen der einzelnen aufeinander folgenden Bilder der Videosequenz. Die eingezeichneten Ellipsen kennzeichnen durch ihren Schwerpunkt den erkannten *space-time interest point*. Die Ausdehnung der Ellipsen in Raum und Zeit stehen für die Skale, auf der die Punkte erkannt wurden. Links: Fußballsequenz; der *interest point* wird beim Kopfball des mittleren Spielers erkannt. Rechts: Sequenz einer klatschenden Person; die Punkte werden erkannt, wenn sich die Handflächen in der Mitte treffen. Aus „on space-time interest-points“ [Lap05]

markiert. Wie zu sehen ist, wurden die Punkte wie erwartet dort erkannt, wo sich die Handflächen treffen.

Die gesuchten Punkte befinden sich ganz allgemein in Regionen, in denen die Bewegung texturierter Bildstrukturen nicht konstant ist. Also dort, wo sich diese Strukturen beschleunigen, verlangsamen oder ihre Richtung ändern. Die folgende Illustration soll dies noch einmal verdeutlichen: Ein Ball fliegt von links gegen eine Mauer und wird reflektiert. Im dritten Bild ist der zeitliche Verlauf dieser Sequenz als x-t Schnitt aufgetragen. Der Punkt höchster Variabilität in zeitlicher Richtung ist der Zeitpunkt, an dem der Ball von der Mauer reflektiert wird und sich seine Bewegungsrichtung umkehrt. Ähnlich zu dem 2D-Beispiel in Abbildung 3.1 auf Seite 11, zeigt sich auch im zeitlichen Verlauf der „Ecken“-Charakter der detektierten Punkte.



Der hier vorgestellte Harris3D Operator detektiert diese Punkte in einer Videosequenz

Abbildung 3.3.: ***Scale space*** **Beispiel. Oben: Durch Tiefpassfilterung mit einem Gaußkern mit** $\sigma^2 = (0, 2, 4, 8, 16)$ **werden hochfrequente Strukturen nach und nach verwischt. Unten: Durch Tiefpassfilterung mit einem Gaußkern mit** $\tau^2 = (0, 2, 4, 8, 16)$ **werden hochfrequente Bewegungsanteile nach und nach verwischt.**

ganz allgemein an den Stellen, an denen die Variabilität in Raum und Zeit ein lokales Maximum annimmt. Bildstrukturen, die sich mit konstanter Geschwindigkeit bewegen, werden hingegen nicht detektiert.

3.2.1. Multiskalen Detektion

Bewegungen werden nicht immer in der selben Geschwindigkeit oder im selben Abstand zur Kamera ausgeführt, sie finden in unterschiedlichen räumlichen und zeitlichen Skalen statt. So kann man sich zum Beispiel eine gehende Person vorstellen, die von einer Kamera in großer Entfernung aufgenommen wird. In der Aufnahme wird die Person sehr klein erscheinen. Je näher die Kamera der Person kommt, desto größer wird die Person im Bild dargestellt. Sie erscheint also auf einer größeren räumlichen Skale. Dazu kann die Person natürlich auch schnell oder langsam gehen, was die Ausdehnung eines einzelnen Schrittes auf der Zeitachse verkürzt oder verlängert. Die Bewegungsgeschwindigkeit ist dabei immer die selbe, egal aus welcher Entfernung sie aufgenommen wird. Die räumliche Skale einer Bewegung ist also unabhängig von ihrer zeitlichen Skale. Die Geschwindigkeit der Bewegungsausführung kann darüber hinaus durchaus bedeutungsvoll für die Unterscheidung von verschiedenen Bewegungen sein. Gehen, laufen und rennen lassen sich ohne Berücksichtigung der Geschwindigkeit nur schwer auseinander halten. Aufgrund dieser generellen Unabhängigkeit, müssen die räumlichen und zeitlichen Skalen unterschiedlich behandelt werden.

Gaußscher Skalenraum

Um dieses alltägliche Phänomen in der Bildverarbeitung zu repräsentieren, gibt es das

Konzept des Skalenraums. Der Skalenraum wird zusätzlich zu den intrinsischen Dimensionen (x, y, t) der Video-Bilddomäne $f : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}$ durch die räumliche Skale σ_l^2 und die zeitliche Skale τ_l^2 aufgespannt.

$$L : \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R}_+^2 \to \mathbb{R} \tag{3.4}$$

Die einzelnen Stufen (σ^2, τ^2) im Skalenraum L werden durch Faltung mit einem dreidimensionalen Gaußkern erzeugt.

$$g(x, y, t, \sigma_l^2, \tau_l^2) = \frac{1}{\sqrt{(2\pi)^3 \sigma_l^4 \tau_l^2}} \times exp(-\frac{x^2 + y^2}{2\sigma_l^2 - \frac{t^2}{2\tau_l^2}}) \tag{3.5}$$

Die Parameter σ_l^2 und τ_l^2 stehen für die Varianzen des Filterkerns in räumlicher und zeitlicher Richtung. Der Gaußfilter fungiert hier als optimaler Glättungsfilter, der hochfrequente Details unterdrückt und die groben Strukturen passieren lässt, ohne jedoch fälschlicherweise selbst neue Strukturen zu erzeugen. In ähnlicher Weise verwischen die Details, wenn man den Abstand der Kamera zur Szene erhöht.

Die Bildfunktion an einer bestimmten Position im Skalenraum (σ_l^2, τ_l^2) ergibt sich nun durch Faltung der Video-Bildfunktion f mit dem der Skale entsprechenden Gaußkern g_l.

$$L(\cdot; \sigma_l^2, \tau_l^2) = g_l(\cdot; \sigma_l^2, \tau_l^2) \otimes f(\cdot) \tag{3.6}$$

Zur besseren Lesbarkeit wurden die Positionsparameter (x, y, t) ausgelassen.

3.2.2. Variabilität in Raum und Zeit

Nachdem der Skalenraum aufgespannt ist, muss nun eine fundierte mathematische Definition der Variabilität der Bildfunktionen $L(\sigma^2, \tau^2)$ gefunden werden, deren lokale Maxima auf den gesuchten *interest points* liegen. Der Detektor von Laptev erweitert dabei die analytische Herangehensweise von Harris und Stephens in die raum-zeitliche Domäne. Ähnlich dem Harris Corner Detector (Abschnitt 3.1.2) werden zunächst die partiellen Ableitungen erster Ordnung auf allen Skalen (σ_l^2, τ_l^2) berechnet.

$$\begin{aligned} L_x(\cdot; \sigma_l^2, \tau_l^2) &= \partial_x(g_l \otimes f) \\ L_y(\cdot; \sigma_l^2, \tau_l^2) &= \partial_y(g_l \otimes f) \\ L_t(\cdot; \sigma_l^2, \tau_l^2) &= \partial_t(g_l \otimes f) \end{aligned}$$

Die Momentenmatrix zweiter Ordnung ergibt sich nun an jeder Position $(x, y, t, \sigma_l^2, \tau_l^2)$ im Skalenraum als

$$M = g_i(\cdot; \sigma_i^2, \tau_i^2) \otimes \begin{pmatrix} L_x^2 & L_x L_y & L_x L_t \\ L_x L_y & L_y^2 & L_y L_t \\ L_x L_t & L_y L_t & L_t^2 \end{pmatrix} \tag{3.7}$$

Zur besseren Lesbarkeit wurden die Positionsparameter $(x, y, t, \sigma_l^2, \tau_l^2)$ an der Matrix M und an den partiellen Ableitungen ausgelassen. Zu beachten ist, dass der Gaußkern $g_i(\cdot; \sigma_i^2, \tau_i^2)$ hier ein Integrationsfenster bildet und nicht die Skalen-Repräsentation aufspannt. Die Parameter (σ_i^2, τ_i^2) beschreiben die Größe des gaußschen Integrationsfensters und hängen von den Skalenparametern (σ_l^2, τ_l^2) über den Parameter s ab.

$$\sigma_i^2 = s\sigma_l^2$$
$$\tau_i^2 = s\tau_l^2$$

Die *interest points* liegen jetzt an Positionen von f, die signifikante Eigenwerte $\lambda_1, \lambda_2, \lambda_3$ von M aufweisen. Um diese Positionen möglichst effizient zu finden, hat Laptev vorgeschlagen, die Harris Funktion (Formel 3.3) auf die raum-zeitliche Domäne auszudehnen.

$$H = det(M) - k\, trace^3(M) \tag{3.8}$$
$$= \lambda_1\lambda_2\lambda_3 - k\,(\lambda_1 + \lambda_2 + \lambda_3)^3 \tag{3.9}$$

Die Eigenwerte brauchen so nicht explizit berechnet zu werden, da allein deren Verhältnis zueinander schon genug über den zugrunde liegenden Bildbereich verrät.

Um zu zeigen, dass positive lokale Maxima von H mit Punkten korrespondieren, die hohe Eigenwerte $\lambda_1, \lambda_2, \lambda_3$ $(\lambda_1 \leq \lambda_2 \leq \lambda_3)$ aufweisen, definiert Laptev [Lap05] das Verhältnis $\alpha = \lambda_2/\lambda_1$ und $\beta = \lambda_3/\lambda_1$ und formuliert H als

$$H = \lambda_1^3(\alpha\beta - k(1 + \alpha + \beta)^3).$$

Aus der Forderung $H \geq 0$ ergibt sich $k \leq \alpha\beta/(1 + \alpha + \beta)^3$ und es folgt, dass k sein Maximum $(k = 1/27)$ bei $\alpha = \beta = 1$ annimmt. Für ausreichend große Schwellwerte k korrespondieren positive lokale Maxima von H mit hoher Variabilität der Bildintensitäten in räumlicher und zeitlicher Richtung.

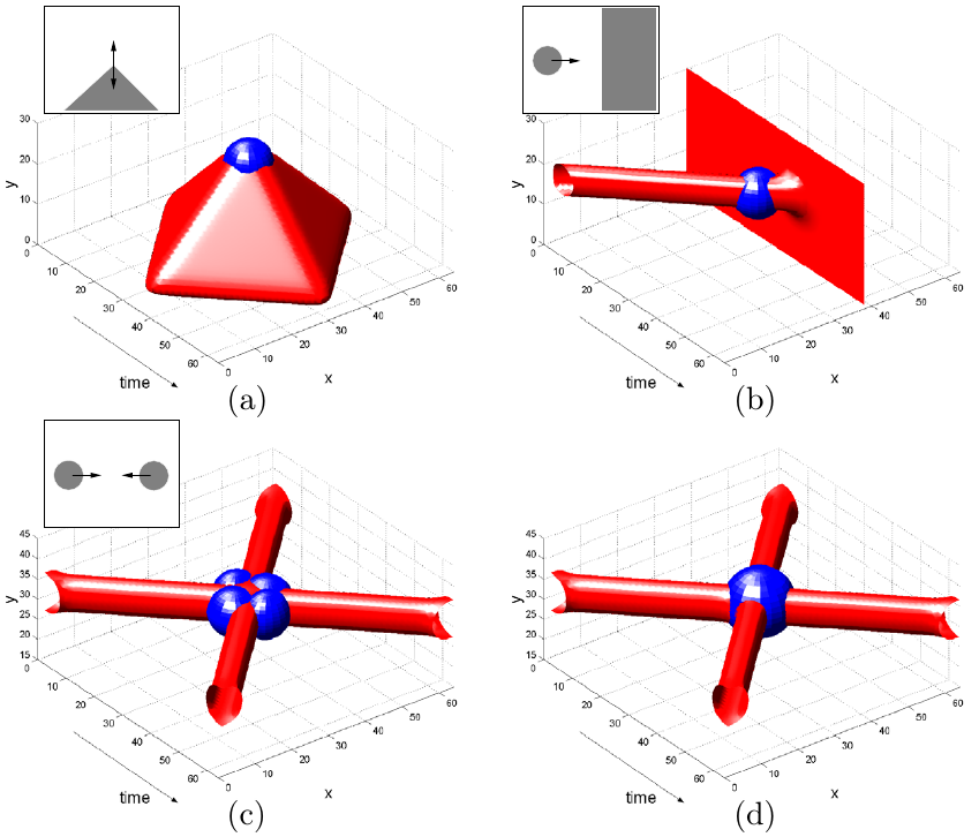


Abbildung 3.4.: Drei Raum-Zeit-Plots synthetischer Bewegungssequenzen. (a) Ein sich von unten nach oben und wieder zurück bewegendes Dreieck. (b) Verschmelzen eines Balles mit einer Wand. (c) Elastische Kollision zweier Bälle mit Detektion auf der Skale $(\sigma^2 = 8, \tau^2 = 8)$ (d) Die selbe Sequenz mit Detektion auf der Skale $(\sigma^2 = 16, \tau^2 = 16)$. Aus „on space time interest points“ [Lap05].

3.2.3. Analyse der detektierten Punkte

Der Harris 3D Detektor reagiert, wie eingangs erwähnt auf „Ecken“ in Raum und Zeit. Welche Raum-Zeit-Strukturen darunter fallen und wie sie detektiert werden, soll zunächst an synthetischen Beispielen illustriert werden. Abbildung 3.4 zeigt Raum-Zeit-Plots von drei synthetischen Sequenzen. Oben links ist eine Sequenz zu sehen, auf der sich ein gefülltes Dreieck zuerst von unten nach oben ins Bild schiebt, in der Mitte die Bewegung umkehrt und unten wieder aus dem Bild verschwindet. Die blaue Kugel markiert den an der Stelle detektierten *interest point*, an der sich die Bewegung umkehrt. Die nächste Sequenz zeigt einen Ball, der von links auf eine Mauer trifft. Der *interest point* wird auch hier wieder in dem Moment erkannt, in dem sich die Bewegung umkehrt. Die dritte Sequenz zeigt zwei Bälle, die aufeinander prallen.

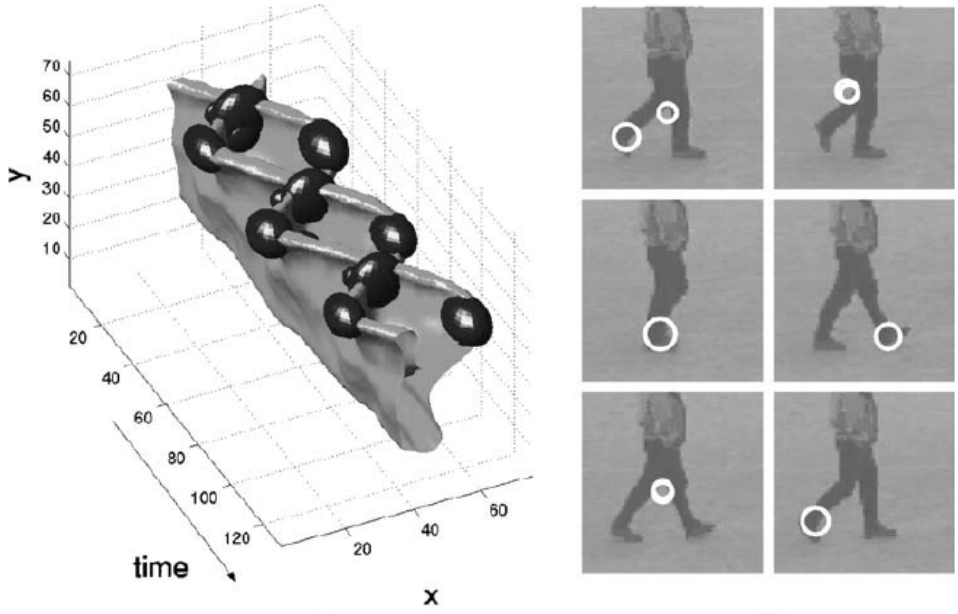


Abbildung 3.5.: Schwellwert-Oberfläche der Beinbewegung und Detektionen in der zugehörigen Videosequenz. Die Oberfläche links zeigt die Beinbewegungen der Person aus dem Video auf dem Kopf stehend, die Füße sind oben als Bergkämme zu sehen. Die Kugeln in dem Plot entsprechen den erkannten *interest points* aus der Sequenz rechts. Es ist gut zu erkennen, wie nicht konstante Bewegungen zu „Ecken" im Raum-Zeit-Verlauf führen. Aus „on space time interest points" [Lap05].

Die Detektionen sind für zwei verschiedene Skalen $\sigma^2 = 8, \tau^2 = 8$ und $\sigma^2 = 16, \tau^2 = 16$ eingezeichnet. Es zeigt sich, dass auf kleiner Skale (links) die Detektion wesentlich feiner aufgelöst ist und die einzelnen Phasen des Verschmelzens beider Bälle erkannt werden.

Ecken in Raum und Zeit

Abbildung 3.5 zeigt eine Laufsequenz, die mit einer stabilisierten, mitfahrenden Kamera aufgenommen wurde. Die Person ist hier immer in der Mitte des Videobildes zu sehen. Links ist eine Schwellwert-Oberfläche aufgetragen. Die Abbildung ist um 180° gedreht, die Füße sind im Diagramm oben. Rechts sind Ausschnitte aus der zugehörigen Videosequenz. Hier ist besonders gut zu sehen, was das Kriterium der nicht konstanten Bewegung bzw. der „Ecken" in Raum und Zeit konkret bedeutet. Die Punkte werden jeweils erkannt, wenn die Füße ihre Extremposition erreicht haben und die Bewegungsrichtung umdreht. In der Abbildung ist zu sehen, dass der zeitliche Verlauf der Fußbewegung dort eine „Ecke" zeichnet. Der Nulldurchgang der Fußbewegungen wird ebenfalls erkannt, da sich auch hier die Richtung der Bewegung umdreht, ähnlich der Sequenz der beiden voneinander abprallen-

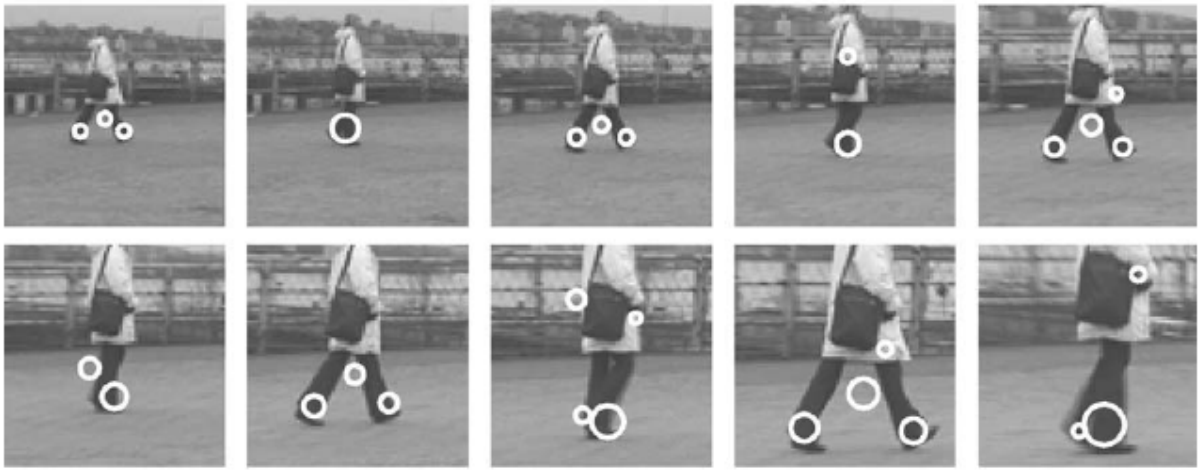


Abbildung 3.6.: Oben: Detektierte Punkte auf einer Zoom-Sequenz einer laufenden Person. Die räumliche Skale der Detektion ist durch die Größe der Kreise um den detektierten Punkt angegeben. Die Detektionsskalen korrespondieren gut mit der größer werdenden räumlichen Ausdehnung der Bildstrukturen (in diesem Fall die Beine). Aus „on space-time interest-points“ [Lap05].

den Bälle. Wenn man den Kamm des Raum-Zeit-Plots in der x-t Ebene betrachtet, sieht man die „Ecken“ im zeitlichen Verlauf der Fußbewegung.

Skalenänderungen

Abbildung 3.6 zeigt eine weitere Sequenz einer gehenden Person. Die Kamera fährt die Seitwärtsbewegung wieder mit, zoomt jedoch langsam in die Szene hinein. Die räumliche Skale der Detektionen sollte also zunehmen. Die zeitliche Skale bleibt unverändert, da die Frequenz der Bewegung nicht variiert wird. Die erkannten *interest points* sind in der Abbildung durch Kreise markiert. Der Radius spiegelt die räumlich Skale der Detektion wider und nimmt wie erwartet zu je weiter die Kamera in die Szene hineinzoomt. Der Detektor zeigt sich also weitgehend robust gegenüber Skalenänderungen.

Vergleich mit rein spatialer Detektion

Abbildung 3.7 vergleicht die raum-zeitlichen *interest points* des Harris3D Operators mit den Detektionen des rein spatial arbeitenden Harris2D Operators. Auffallend ist, dass der raum-zeitliche Operator deutlich selektiver ist und insbesondere besser auf für die Bewegungsklassifikation interessante Primitive reagiert. Die Anzahl an Detektionen im Hintergrund sind deutlich geringer.

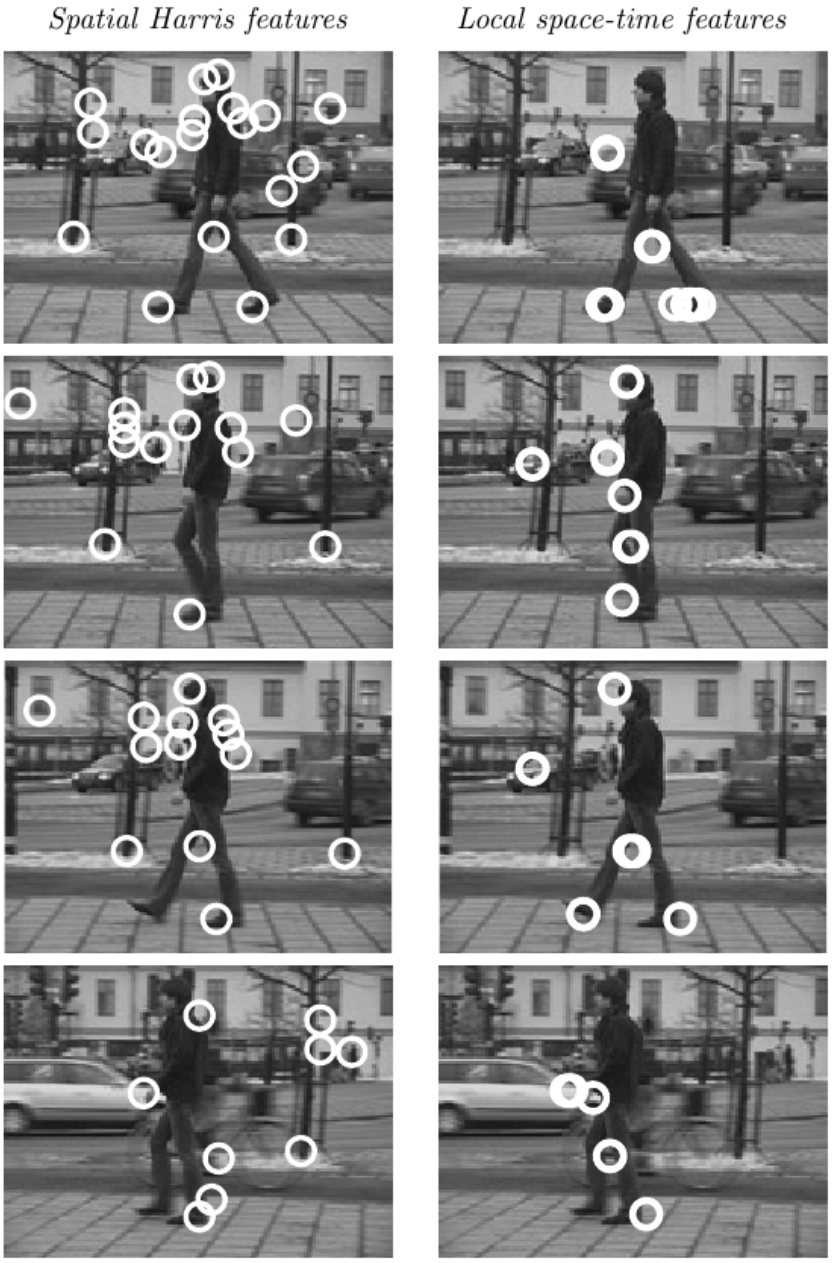


Abbildung 3.7.: Vergleich lokaler *interest points*. Links: Maxima des räumlichen Harris2D Operators. Rechts: Maxima des raum-zeitlichen Harris3D Operators. Für den Vergleich wurde etwa die selbe Anzahl der jeweils stärksten detektierten Punkte für beide Methoden eingezeichnet. Der Harris3D Detektor kann mehrere Detektionen auf dem selben Pixel aber in unterschiedlichen Skalen erfassen. Zu sehen ist, dass der raum-zeitliche Operator selektiver ist und vor allem Bildstrukturen bevorzugt, die nicht konstanter Bewegung unterliegen. Solche Punkte sind typisch für die ausgestreckte Beinbewegung. Andere Punkte werden detektiert, wenn Bildstrukturen des Hintergrundes verdeckt werden. Aus „Local Spatio-Temporal Image Features for Motion Interpretation“ [Lap04].

3.3. Repräsentation als Bag of Visual Words

Durch die Detektion der *space time interest points* sind nun Positionen im Video gefunden, die mit hoher Wahrscheinlichkeit interessante Informationen über die zu klassifizierenden Aktionen bereithalten. Bisher sind zu den *interest points* jedoch nicht mehr als ein Tupel aus der Position (x, y, t) in der Raum-Zeit und der räumlichen und zeitlichen Skale (σ^2, τ^2) bekannt. Es müssen weitere Informationen aus der Umgebung der erkannten Punkte extrahiert werden, um später sicher zwischen den verschiedenen Aktionen unterscheiden zu können.

3.3.1. Gradienten und optischer Fluss als Deskriptoren

Um die Nachbarschaften der erkannten *interest points* zu charakterisieren, sieht das Verfahren zwei Arten von Deskriptoren vor. Der HoG Deskriptor, *histogram of oriented gradients*, wird als Histogramm über den Winkeln der spatialen Gradienten berechnet. Damit werden vor allem Informationen über die Textur in der Umgebung des zu repräsentierenden *interest points* kodiert. Der HoF Deskriptor, *histogram of optical flow*, wird als Histogramm über die Bewegungsrichtungen von Bildstrukturen im Video berechnet. Hierdurch werden vor allem Informationen über den Bewegungsverlauf in der Umgebung erfasst. Die genaue Berechnung der Deskriptoren wird in den Abschnitten 4.2.2 und 4.2.3 erläutert.

3.3.2. Das Bag of Visual Words Modell

Das *bag of words* Modell stammt ursprünglich aus der Dokumentenklassifikation. Die *bag* Semantik steht dabei generell für eine ungeordnete Liste, wie ein Sack, in den man die einzelnen Elemente einfach hinein gibt und damit jegliche Ordnung der Elemente aufhebt. Ein Dokument wird hierbei als solch ein *bag* der einzelnen Wörter dargestellt, bei dem nur noch die Frequenz bekannt ist, mit der die Wörter aufgetreten sind, ihre Position im Text jedoch nicht. Bei geschriebener Sprache ist das relativ einfach, da es schon ein definiertes Vokabular gibt. Die aus den Videodaten extrahierten HoG und HoF Deskriptoren haben jedoch keine intrinsische Bedeutung, die ein definiertes Vokabular vorgeben würde. Um das *bag of words* Modell dennoch anwenden zu können, muss ein Vokabular an „visuellen“ Wörtern aufgebaut werden, so dass jeder mögliche HoG und HoF Deskriptor eine Entsprechung in einem Wort des Vokabulars hat. Dafür muss ein Ähnlichkeitsmaß gefunden werden, mit dem unterschiedliche Ausprägungen der Deskriptoren eines Wortes zusammengefasst werden können. Es müssen also ähnliche visuelle „Sprechweisen“ der Wörter aufeinander abgebildet werden. Das hier beschriebene Verfahren zum Aufbau des

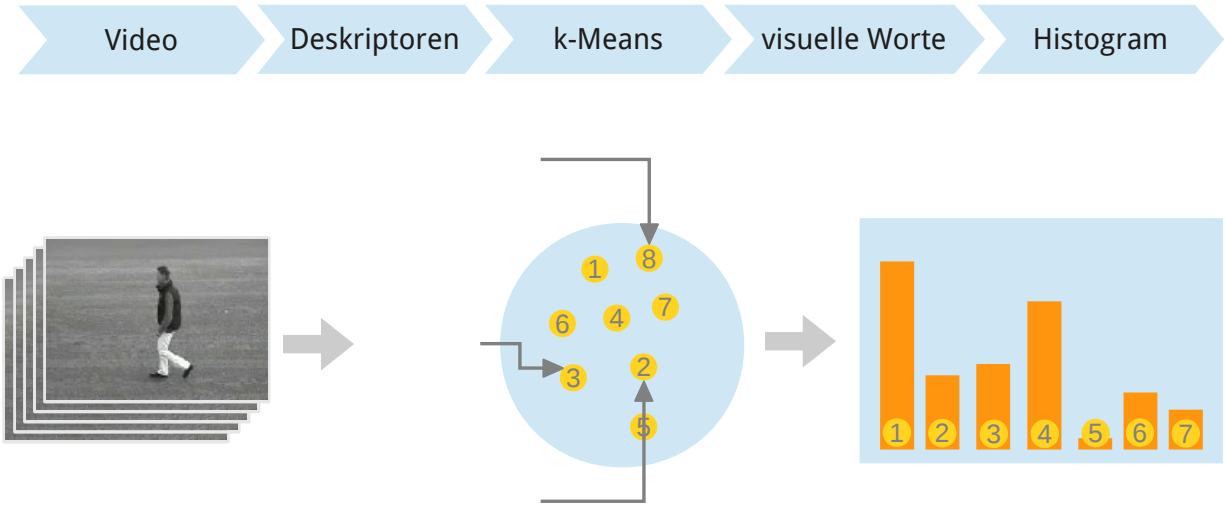


Abbildung 3.8.: Ablaufdiagramm des *bag of visual words* Modells. Aus der Videosequenz werden auf den detektierten *interest points* Deskriptoren extrahiert. Diese werden mit k-Menas nach Ähnlichkeit gruppiert und durch die ID ihres jeweiligen Clusters ersetzt. Diese IDs werden im *bag of visual words* Histogramm aufgetragen.

bag of visual words Modells basiert auf dem Paper „Learning Realistic Human Actions from Movies“ [LMSR08] von Laptev, Marszałek, Schmid und Rozenfeld.

3.3.3. Aufbau des Vokabulars

Um das visuelle Vokabular aufzubauen, werden zuerst die HoG und HoF Deskriptoren des selben Punktes zu einem gemeinsamen *feature* Vektor zusammengefasst. Auf diesen *feature* Vektoren werden dann mit dem k-Means Algorithmus k Cluster ermittelt. Jedes der ermittelten Cluster bekommt eine eindeutige Nummer, die für eines der k Wörter des Vokabulars steht. Zu jedem *feature* Vektor wird nun das am nächsten gelegene Clusterzentrum nach euklidscher Distanz ermittelt und der Vektor wird durch die Nummer des Clusters ersetzt. Alle durch die Clusternummern repräsentierten *feature* Vektoren einer Videosequenz werden nun in ein gemeinsames Histogramm eingetragen. Die gesamte Sequenz wird also auf ein Histogramm über alle darin vorkommenden visuellen Wörter reduziert. Um die Histogramme unterschiedlicher Sequenzen vergleichbar und unabhängig von der Anzahl der extrahierten Deskriptoren zu machen, werden sie nach L_1 Norm

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i|$$

normalisiert. Das Histogramm bildet somit einen k dimensionalen *feature* Vektor für die nachfolgende Klassifikation.

3.4. Aktionsklassifikation

Die Klassifikation der Videosequenzen erfolgt nun auf der *bag of visual words* Repräsentation der Videosequenzen. Jede Videosequenz ist durch ein normalisiertes Histogramm über alle in der Sequenz vorkommenden visuellen Worte repräsentiert. Das Histogramm enthält für jedes der insgesamt k Wörter des Vokabulars einen Eintrag. Zur Klassifikation wird bei diesem Verfahren eine nicht-lineare *support vector machine* eingesetzt, deren Funktionsweise im Folgenden kurz dargestellt werden soll.

3.4.1. Support Vector Machine

Die *support vector machine*, kurz SVM, ist ein binärer *large margin classifier*, der versucht, eine optimale Trennhyperebene zwischen Beispielen zweier Klassen zu finden. Die SVM legt dabei besonders Wert auf eine möglichst gute Generalisierung. Die Trainingsbeispiele sollen also nicht nur optimal „auswendig“ gelernt werden, sondern neue Beispiele sollen treffsicher der richtigen Klasse zugeordnet werden, auch wenn sie leicht unterschiedlich sind.
Um das zu verdeutlichen, sollen zunächst zwei zu unterscheidende Klassen, die Kreise und die Rechtecke, betrachtet werden. Diese Klassen liegen auf einer zweidimensionalen Ebene und sollen durch eine lineare Trennfunktion, im Zweidimensionalen eine Gerade, separiert werden. Wie die Illustration zeigt gibt es in diesem Fall, in dem die Klassen gut linear

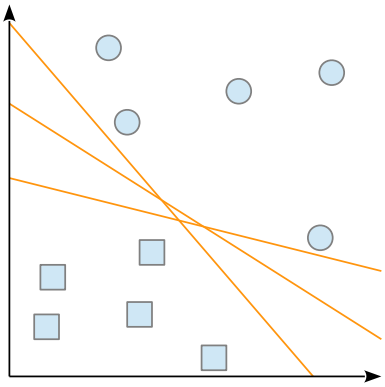


Abbildung 3.9.: Es gibt viele Möglichkeiten eine Trenngerade zwischen zwei linear separierbaren Klassen einzuziehen.

trennbar sind, prinzipiell unendlich viele Möglichkeiten, die Trenngerade einzuziehen. Es muss also ein Kriterium gefunden werden, das die Güte der Separation bewertet.

Abstand der Trennhyperebene zu den Beispielen

Die Trennung ist offensichtlich schlecht, wenn die Trenngerade zu nah an einem der Beispiele vorbei führt, da die Generalisierung darunter leiden würde. Deshalb wird versucht, die Trenngerade so weit wie möglich von den Trainingsbeispielen entfernt einzuziehen. Hierbei ist es nicht notwendig, alle Trainingsbeispiele zu betrachten, es werden nur Beispiele einbezogen, die der Trenngeraden am nächsten liegen. Diese Vektoren werden *support vectors* genannt, nach denen die *support vector machine* benannt ist. Der SVM Algorithmus probiert nun diejenige Trennfunktion zu finden, die den größtmöglichen Minimalabstand zu den Trainingsbeispielen aufweist. In der SVM Terminologie wird dieser Abstand als *margin* bezeichnet.

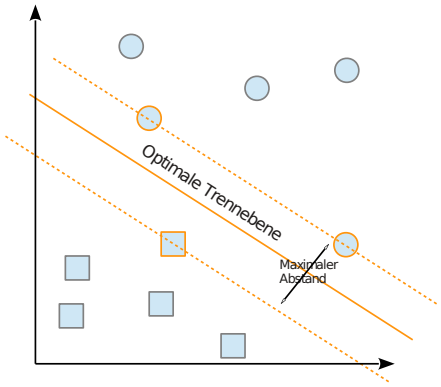


Abbildung 3.10.: Trennfunktion mit maximalem Abstand zu den am nächsten gelegenen Vertretern der beiden Klassen.

Nicht linare Trennung der Klassen

In der Praxis lassen sich jedoch nur die wenigsten Klassenverteilungen linear trennen. Um für diese Verteilungen dennoch Trennfunktionen zu finden, werden die Daten in einen höher dimensionalen Raum projeziert, indem sie sich annähernd linear trennen lassen. Auf diese Weise lässt sich auf mathematisch schwieriger zu handhabende, nicht lineare Trennfunktionen wie z.B. Parabeln verzichten. Um diese Transformation möglichst effizient auszuführen, bedient man sich des sogenannten Kernel-Tricks.
Die mathematische Grundlage dieses Tricks ist dabei der Satz von Mercer. Der Satz besagt, dass es für einen Datensatz X und eine Kernfunktion $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ eine Abbildung $\phi : \mathbb{R}^d \to \mathbb{R}^D$ gibt, sodass

$$k(x_j, x_k) = \phi(\hat{x}_j) * \phi(\hat{x}_k)^T \tag{3.10}$$

Eine Abbildung von X auf $\hat{X}$ kann also dadurch erfolgen, dass Skalarprodukte in X durch Kernfunktionen in X ersetzt werden. Die explizite Ausführung der Transformation von X nach $\hat{X}$ ist nicht nötig.

Mehrklassenprobleme

Die SVM ist von Ansatz her nur für 2-Klassen Probleme ausgelegt. Es gibt jedoch mehrere Erweiterungen, die die Funktionsweise auf mehrere Klassen ausweiten. Prinzipiell lässt sich auch jedes N-Klassen Problem in $(N-1)$ 2-Klassen Probleme aufteilen, indem man jeweils eine Klasse gegen alle anderen oder eine Klasse gegen den noch nicht ausgeschlossenen Rest abgrenzt. In diesem Verfahren wird der „einer gegen alle“ Ansatz verwendet.

Kapitel 4

Implementation des offline Aktionsklassifikators

Im Rahmen dieser Arbeit wurden alle drei Phasen des Aktionsklassifikators, wie in Abbildung 4.1 nochmals illustriert, umgesetzt. Der hier implementierte Detektor orientiert sich nah an den Ideen in dem Paper „On Space-Time Interest Points“ [Lap05] von Ivan Laptev. Zur Evaluation der detektierten Punkte wird die *closed source* Referenzimplementation von Laptev[Lap13] herangezogen. Viele Details sind jedoch undokumentiert und mussten experimentell ermittelt werden. Daher kann nicht garantiert werden, dass der hier implementierte Detektor der Implementation von Laptev in jedem Detail entspricht. Es haben sich jedoch beide Detektoren als gleich mächtig in Bezug auf die spätere Klassifikation erwiesen.

4.1. Detektion der Space Time Interest Points

Der *space time interest point* Detektor wurde komplett in C++11 implementiert und nutzt als einzige Abhängigkeit die Bildverarbeitungsbibliothek *opencv*, die 1999 von Intel initiiert wurde. Ein wichtiges Designziel bei der Implementation des Detektors war eine hohe Verarbeitungsgeschwindigkeit, um auch Live-Bilder in ausreichend hoher Auflösung flüssig verarbeiten zu können. Um dieses Ziel zu erreichen, ist das Programm von Anfang an auf Multithreading und die Nutzung mehrerer Rechenkerne ausgelegt. Der Detektor arbeitet intern auf 32 Bit Gleitkomma Grauwertbildern um, wenn möglich, auf die SIMD Erweiterungen der Prozessoren zurückgreifen zu können. Eingehende Videobilder werden zunächst konvertiert.

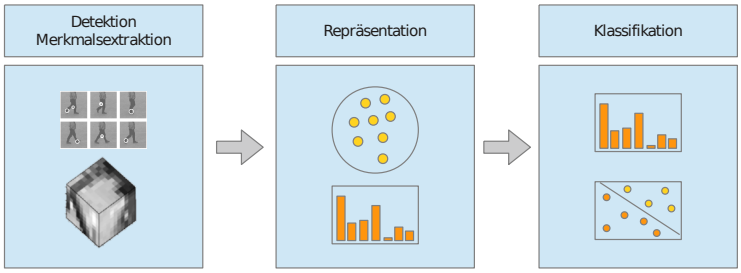


Abbildung 4.1.: Schematischer Ablauf des Aktionsklassifikators nach Laptev. Zunächst werden *space-time interest points* detektiert in deren Nachbarschaften HoG und HoF Deskriptoren extrahiert werden. Die zu klassifizierende Videosequenz wird als *bag of visual words* repräsentiert und mit einer SVM klassifiziert.

Systemdesign

Das Systemdesign gliedert sich in zwei Schichten. Die erste Schicht kontrolliert und unterstützt den Ablauf der Detektion von der Akquise und Aufbereitung der Videobilder bis zur Integration der Detektionsergebnisse. Dazu gehören folgende Schritte:

- Laden und konvertieren eingehender Videobilder
- Vorbereitung der Bildrepräsentation im Skalenraum
- Koordinierung der Detektion auf den einzelnen Skalenebenen
- Integration der Ergebnisse der einzelnen Skalen

In der zweiten Schicht findet sich für jede Kombination der Skalenparameter (σ^2, τ^2) jeweils ein eigener Detektor. Jeder dieser Detektoren berechnet die Positionen der *interest points* unabhängig von den Detektoren auf den anderen Skalen. Der Prozess der Detektion gliedert sich grob in die folgenden Schritte:

- Transformation des eingehenden Bildes in die zu der jeweiligen Skale gehörenden Repräsentation
- Berechnen der Momentenmatrix und der Harris3D Funktion (S. 17, Formel 3.9)
- Bereitstellen der *interest points* als lokale Maxima der Harris3D Funktion

4.1.1. Aufbau des Skalenraums

Die *scale-space* Repräsentation wird in zwei Stufen erzeugt. Zuerst wird aus einem eingehenden Videobild eine Skalenpyramide berechnet, deren unterste Stufe aus dem Originalbild besteht und jede weitere Stufe das Bild sukzessive stärker gefiltert und in geringerer Auflösung enthält.

Ausgehend von dem Originalbild wird jede weitere Stufe berechnet, indem die vorhergehende Stufe zuerst mit einem Gaußkern $\sigma_l^2 = 2$ gefiltert wird. Danach wird jede zweite Bildzeile und Spalte gestrichen, um die Auflösung entlang der beiden Bilddimensionen zu halbieren. Jede Pyramidenstufe ρ wird jetzt in allen 4 Kombinationen der Skalenparameter $\sigma_l^2 \in \{4, 8\}$ und $\tau_l^2 \in \{2, 4\}$ jeweils von eigenen Detektoren weiterverarbeitet. Mit den voreingestellten Pyramidenstufen $\rho \in \{1, 2, 3\}$ ergeben sich insgesamt also 12 unterschiedliche Skalen in Raum und Zeit, auf denen *interest points* detektiert werden.

$$\rho \in \{1, 2, 3\} \times \sigma_l^2 \in \{4, 8\} \times \tau_l^2 \in \{2, 4\}$$

Verarbeitung der einzelnen Skalenebenen

In jeder dieser Skalenstufen beginnt die Verarbeitung mit dem vorgefilterten Bild aus der entsprechenden Ebene ρ der Skalenpyramide. Dieses Bild wird im ersten Schritt mit einem Gaußkern der zu der aktuellen Skala gehörenden Varianz σ_l^2 gefiltert. Das Resultat wird in einen Bildpuffer geschrieben, der jeweils die letzten N_{τ_l} Bilder vorhält. N_{τ_l} ist die Größe des diskreten temporalen Filterkerns, der im nächsten Schritt zur Anwendung kommt und berechnet sich üblicherweise als

$$N_\tau(\sigma^2) = 6 * \sqrt{\sigma^2} + 1. \tag{4.1}$$

Die Größe des Filterkerns ist dabei so gewählt, dass der Fehler durch die nicht berücksichtigten Randbereiche der Funktion minimal ist. In dem unten stehenden Diagramm mit $\sigma^2 = 4$ ergäbe sich ein Intervall von [-6; 6], was den relevanten Teil der Funktion für eine akkurate Filterung enthält.

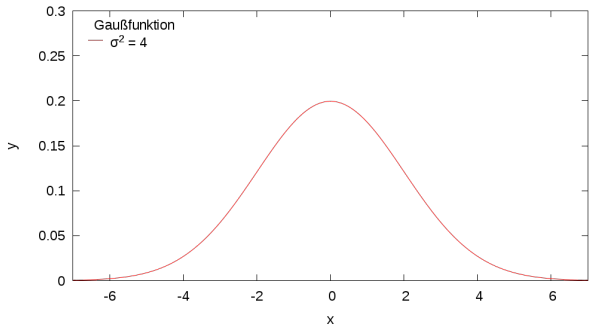


Abbildung 4.2.: Gaußkern $\sigma^2 = 4$

Der Puffer mit den räumlich gefilterten Bildern wird nun in zeitlicher Richtung t mit einem Gaußkern der Varianz τ^2 gefiltert. Da das Ergebnis nur am Ankerpunkt[3]

$$t = \frac{N_\tau}{2}$$

benötigt wird, kommt man hier mit nur einer Multiplikation pro Pixel und ohne Randbetrachtung aus. Dabei ist zu beachten, dass das Bild am Ankerpunkt in der Mitte des Puffers um t Videobilder gegenüber dem aktuell in den Detektor gegebenen Videobild zurückliegt. Die weitere Berechnung erfolgt also mit einer zeitlichen Verzögerung $delay_t = t$.

Das resultierende räumlich und zeitlich gefilterte Bild, mit den *scale-space* Koordinaten $(\rho * \sigma^2, \tau^2)$, wird nun in einen zweiten Puffer geschrieben. Aus dem ersten Puffer wird das zweite Bild entnommen und die räumlichen Gradientenmatrizen L_x und L_y mit einem Sobel-Filter berechnet. Der zeitliche Gradient L_t wird durch Faltung der ersten drei Bilder im zweiten Puffer mit dem Ableitungskern $(0.5; 0; -0.5)$ berechnet. Der Ankerpunkt, auf dem der Gradient berechnet wird, liegt auf dem zweiten gepufferten Bild. Damit erhöht sich die Verzögerung um eins auf $\frac{N_{\tau_l}}{2} + 1$.

4.1.2. Berechnung der Momentenmatrix

Im folgenden Schritt wird die 3x3 Momentenmatrix zweiter Ordnung (3.7) ähnlich wie im 2D-Fall berechnet. Hierzu werden zunächst die Koeffizientenmatrizen L_x^2, L_xL_y, L_xL_t, ..., L_t^2 berechnet. Die Matrizen werden nun mit einem Gaußkern σ_i^2 gemittelt. Die Integrationsskalen ergeben sich als $\sigma_i^2 = s\sigma_l^2$ und $\tau_i^2 = s\tau_l^2$ aus den lokalen Skalen σ_l^2 und τ_l^2. Die geglätteten Koeffizientenmatrizen werden wieder in jeweils eigene Bildpuffer geschrieben, um sie anschließend mit einem Gaußkern τ_i^2 in zeitlicher Richtung zu mitteln. Die Breite des Filterkerns N_{τ_i} berechnet sich wie in 4.1. Der Ankerpunkt liegt wieder bei $N_{\tau_i}/2$. Die Verzögerung zu dem aktuellen Videobild erhöht sich entsprechend.

Aus den räumlich und zeitlich gemittelten Gradientenmatrizen wird nun nach 3.9 die Operatorantwort des Harris3D Detektors berechnet. Die resultierende Matrix H wird ein letztes Mal in einen Puffer der Länge 3 gegeben, so dass die *interest points* durch *non maxima suppression* als lokale räumliche und zeitliche Maxima detektiert werden können. Hierbei

[3]Der Ankerpunkt ist das zentrale Pixel in der Mitte des diskreten Filterkerns. Um immer eine definierte Mitte zu erhalten, werden Filterkerne so konstruiert, dass sie eine ungerade Länge haben.

Parameter	Voreinstellung	Bedeutung
k	5^{-4}	Faktor in H
ϵ	1^{-9}	Minimale Stärke für *interest point* ($H \geq \epsilon$)
s	2	Faktor zwischen Skalen und Integrations-Varianzen
ρ	{1,2,3}	Pyramidenstufen
sPatchSizeFactor	9	Größe der räumlichen Nachbarschaft
tPatchSizeFactor	4	Größe der zeitlichen Nachbarschaft
sSubdevision	3	Anzahl der räumlichen Unterteilungen der Nachbarschaft
tSubdevision	2	Anzahl der zeitlichen Unterteilungen der Nachbarschaft

Tabelle 4.1.: Parameter des Detektors und ihre Bedeutung

werden nur Punkte betrachtet, die über einem Grenzwert ϵ liegen. Die gesamte Verzögerung zum aktuell in den Detektor gegebenen Videobild beträgt am Ende der Berechnung nun

$$delay_t = \frac{N_{\tau_l} + N_{\tau_i}}{2} + 2 \tag{4.2}$$

4.1.3. Integration der Ergebnisse

Dadurch, dass die *scale-space* Repräsentation in zeitlicher Richtung mit zwei verschiedenen Varianzen $\tau_l = 2, 4$ aufgebaut wird, ergeben sich nach Formel 4.2 auch zwei verschiedene Detektorlatenzen. Die detektierten *interest points* werden gesammelt, bis die Ergebnisse aller Skalenstufen vorliegen. Mehrere Detektionen auf dem selben Pixel sind möglich und legitim, da die berechneten Deskriptoren aufgrund unterschiedlicher Nachbarschaften im Skalenraum ebenfalls unterschiedlich sind.

4.2. Deskriptoren

Die detektierten *interest points* alleine sagen noch nicht viel über die Bewegung aus, die sie repräsentieren sollen. Sie liegen allerdings aufgrund des Variabilitätskriteriums der Detektion in Nachbarschaften, die mit hoher Wahrscheinlichkeit interessante Informationen über die Bewegung und ihren Typ enthalten (Abbildung 4.3). Um diese Informationen in einem numerischen Wert zu erfassen, werden an den detektierten Punkten zwei Arten von Deskriptoren berechnet. Der HoG Deskriptor (*histogram of oriented gradients*) erfasst die (x, y) Gradienten des Raum-Zeit Volumens in der betrachteten Nachbarschaft und betrachtet damit eher die Textur der Bilddaten. Der HoF Deskriptor (*histogram of optical*

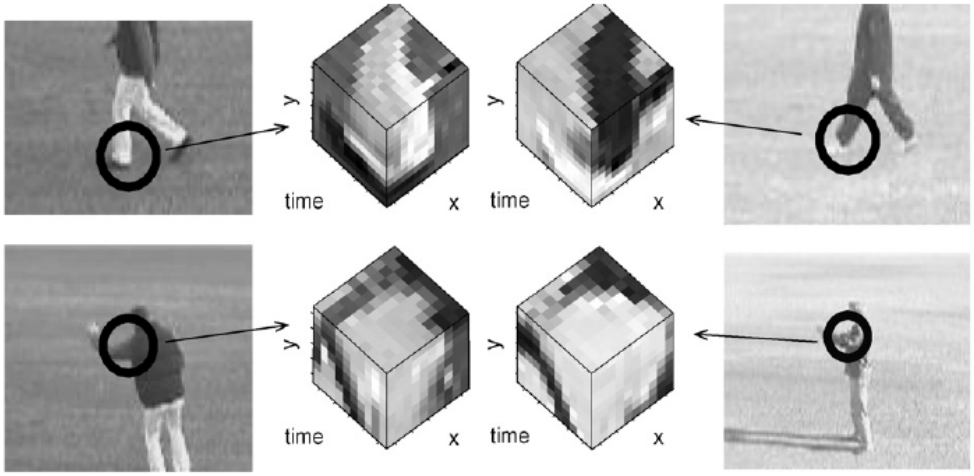


Abbildung 4.3.: An den detektierten interest points werden die lokalen Nachbarschaften extrahiert. In den Abbildungen sind charakteristische Nachbarschaften für „laufen“ und „boxen“ aus jeweils zwei verschiedenen Videos der selben Aktionsklasse zu sehen. Nachbarschaften der selben Aktionsklasse weisen trotz unterschiedlicher Akteure mit unterschiedlicher Kleidung hohe Ähnlichkeit auf. Quelle: [LCSL07]

flow) quantisiert den optischen Fluss in einem Histogramm und kodiert damit eher die lokale Bewegung in der betrachteten Nachbarschaft.

4.2.1. Unterteilung des Raum-Zeit Volumens

Die Größe der Nachbarschaft richtet sich nach dem Skalenlevel, auf dem der *interest point* detektiert wurde. Die Größe wird berechnet als

$$sPatchSize = 2 * sPatchSizeFactor * \sigma \tag{4.3}$$
$$tPatchSize = 2 * tPatchSizeFactor * \tau \tag{4.4}$$

Dieses Volumen wird gleichmäßig in $sSubdevision \times sSubdevision \times tSubdevision$ Untervolumen aufgeteilt, auf denen die Deskriptoren einzeln berechnet werden. Die Histogramme werden anschließend konkateniert und bilden den *feature* Vektor für die Klassifikation.

4.2.2. HoG Deskriptor

Der HoG Deskriptor ist implementiert, indem er für jedes in den Detektor eingehende Bild aus den bereits berechneten spatialen Ableitungen L_x und L_y auf der aktuellen Skale die

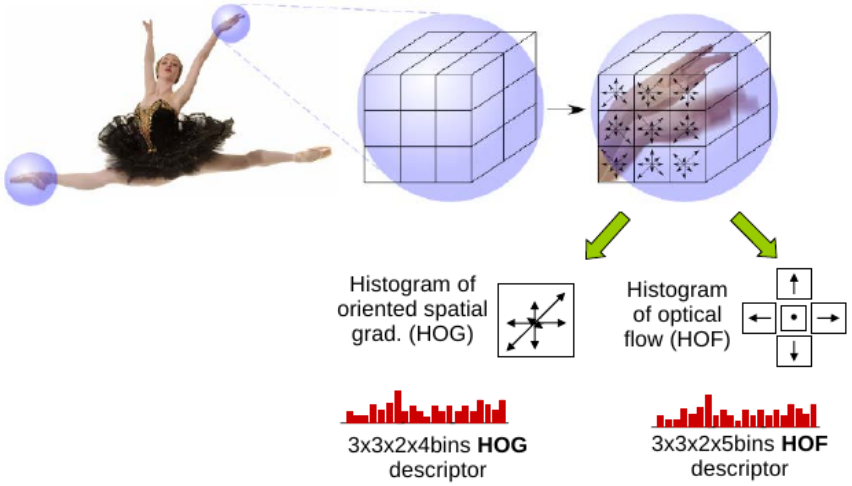


Abbildung 4.4.: Berechnung der HoG und HoF Deskriptoren. Aus einem Vortrag von I. Laptev IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008) [Lap]

Winkel α der Gradienten berechnet.

$$\alpha = atan2(L_x, L_y) \tag{4.5}$$

Die errechneten Winkel werden gepuffert und mit dem Zeitstempel der Ableitungen versehen. Soll zu einem neu detektierten *interest point* der HoG Deskriptor berechnet werden, wird um diesen Punkt eine Nachbarschaft von $sPatchSize \times sPatchSize \times tPatchSize$ aus dem Winkelpuffer extrahiert. Aufgrund der Verzögerung durch die verschiedenen Puffer im Detektor liegen immer mindestens $tPatchSize$ Matrizen mit den Gradientenwinkeln in Vergangenheit und Zukunft um den detektierten Punkt vor. Die extrahierte Nachbarschaft wird wie zuvor beschrieben unterteilt und für jedes Subvolumen werden die Winkel in ein uniformes Histogramm[4] mit 4 *bins* eingeteilt. Die Histogramme über alle Subvolumen werden zu einem gemeinsamen Vektor konkateniert und nach euklidischer Norm normalisiert. Durch die Normalisierung lassen sich Merkmalsvektoren verschiedener Deskriptoren so konkatenieren, dass alle Teilvektoren numerisch mit vergleichbarem Gewicht in den resultierenden Gesamtvektor eingehen.

4.2.3. HoF Deskriptor

Der HoF Deskriptor berechnet zu jedem neu eingehenden Bild auf der aktuellen Skale den optischen Fluss zum vorhergehenden Bild. Der optische Fluss wird nach der Methode von

[4]Uniformes Histogramm meint, dass jeder *bin* die gleiche Ausdehnung hat. In diesem Fall wird der Wertebereich der Winkel von 360° so verteilt, dass jeder *bin* einen Bereich von 90° abdeckt.

Lucas und Kanade berechnet. Da die Flussrichtung ohnehin in ein Histogramm diskretisiert wird, hat es sich als ausreichend erwiesen, den Fluss nur an jedem fünften Pixel zu berechnen, um Rechenzeit einzusparen. Welche Flussschätzung die Originalimplementierung von I. Laptev verwendet, ist nicht dokumentiert. Laufzeitexperimente und Ähnlichkeit der berechneten Deskriptoren unterstützen jedoch die These, dass die Berechnung, wie hier beschrieben, vorgenommen wird. In Abschnitt 5.2 wird sich zeigen, dass die Deskriptoren auch etwa gleich mächtig in Bezug auf die Unterscheidung von Aktionen sind.

Der berechnete optische Fluss wird analog zum HoG Deskriptor in einen Puffer geschrieben. Zu einem detektierten *interest point* wird wieder die $sPatchSize \times sPatchSize \times tPatchSize$ Nachbarschaft um den Punkt extrahiert und in Subvolumen aufgeteilt. Die Flussrichtungen in jedem Subvolumen werden nun auf fünf Histogramm *bins* aufgeteilt. In dieser Implementation wird wieder ein uniformes Histogramm verwendet. Eine Illustration von Ivan Laptev (Abbildung 4.4) lässt jedoch auch den Schluss zu, dass der fünfte Balken alle Pixel enthalten soll, wo keine Bewegung stattgefunden hat, also deren Fluss nahe 0 war. Da die Implementation mit dem uniformen Histogramm jedoch gute Ergebnisse geliefert hat, wurde diese Interpretation nicht weiter verfolgt. Anschließend werden die Histogramme aller Subvolumen wieder zu einem *feature* Vektor konkateniert und nach euklidischer Norm normalisiert. Im Folgenden wird der Prozess der Bewegungsschätzung, wie er hier implementiert ist, näher erläutert.

Lucas-Kanade Methode

Die Lucas-Kanade Methode ist eine weit verbreitete differentielle Herangehensweise an die Bewegungsschätzung. Bewegungsschätzung ist der Prozess, aus Bildsequenzen und den Änderungen der einzelnen Bilder zueinander auf die Bewegungen der abgebildeten Objekte zu schließen. Diese Bewegungen sind aus der Videosequenz jedoch nicht direkt ablesbar. Es lassen sich nur deren Auswirkungen in Form eines optischen Bewegungsfeldes beobachten. Der Psychologe James Gibson prägte dafür um 1950 in seinem Buch „The perception of the visual world“ [Gib50] den Begriff *optical flow*. Um den optischen Fluss zu berechnen, muss man die Annahme treffen, dass jegliche Änderung zwischen den Einzelbildern einer Videosequenz direkt durch Bewegung in der abgebildeten Szene entstanden ist. Diese Einschränkung lässt sich mathematisch als Grundgleichung des optischen Flusses

$$\frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial y}\Delta y = -\frac{\partial f}{\partial t} \tag{4.6}$$

ausdrücken.

Diese Annahme wird jedoch verletzt wenn sich z.B. die Beleuchtungssituation ändert. Eine Lichtquelle kann ihre Intensität, Farbe oder Position verändern und damit große Änderungen an dem Erscheinungsbild der abgebildeten Objekte hervorrufen, ohne dass sich die

Objekte bewegt haben müssen.

Bewegung in der abgebildeten Szene schlägt sich in einer Verschiebung von Bildstrukturen der Abbildung nieder. Der optische Fluss lässt sich durch diese Verschiebungen zwischen zwei Bildern einer Sequenz ausdrücken. Es ist also ein Korrespondenzproblem, welche Bildstruktur in Bild (t-1) an welche Position im darauf folgenden Bild (t) der Videosequenz gewandert ist. Es gibt viele Herangehensweisen, um dieses Problem zu lösen. Im Folgenden soll die Methode von Lucas und Kanade [LK81] vorgestellt werden.

Allein durch Gleichung (4.6) sind die beiden Unbekannten Δx und Δy jedoch unterbestimmt, es werden noch weitere Gleichungen benötigt. Lucas und Kanade führten eine weitere Einschränkung ein, mit der Grundannahme, dass der optische Fluss in der lokalen Nachbarschaft w des aktuellen Pixels p weitgehend konstant ist. Ausgehend von Gleichung (4.6) muss der lokale Fluss $(\Delta x, \Delta y)$ also folgende Gleichungen erfüllen.

$$\begin{aligned} f_x(w_1)\Delta x + f_y(w_1)\Delta y &= -f_t(w_1) \\ f_x(w_2)\Delta x + f_y(w_2)\Delta y &= -f_t(w_2) \\ &\vdots \\ f_x(w_n)\Delta x + f_y(w_n)\Delta y &= -f_t(w_n) \end{aligned}$$

Wobei $w_{(1\ldots n)}$ alle Positionen aus dem Nachbarschaftsfenster w bezeichnen; f_x, f_y, f_t sind die partiellen Ableitungen der Bildfunktion f an der Position p. Diese Gleichungen lassen sich in Matrixform schreiben $Av = b$ mit

$$A = \begin{bmatrix} f_x(w_1) & f_y(w_1) \\ f_x(w_2) & f_y(w_2) \\ \vdots & \vdots \\ f_x(w_n) & f_y(w_n) \end{bmatrix}, \quad v = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}, \quad b = \begin{bmatrix} -f_t(w_1) \\ -f_t(w_2) \\ \vdots \\ -f_t(w_n) \end{bmatrix} \tag{4.7}$$

Da dieses System nun mehr Gleichungen als Unbekannte hat, ist es überbestimmt. Die Lucas-Kanade Methode versucht die Lösung jetzt mit der Methode der kleinsten Quadrate zu nähern.

$$A^T A v = A^T b \tag{4.8}$$

oder aufgelöst nach v

$$v = (A^T A)^{-1} A^T b \tag{4.9}$$

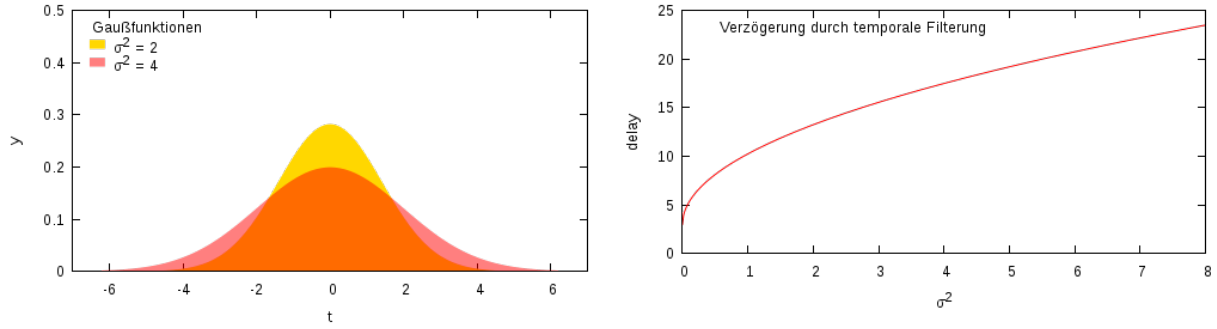


Abbildung 4.5.: Verzögerung in der Detektion durch die Filterung in zeitlicher Richtung zum Aufbau des Skalenraums

Die Matrix $A^T A$ ist die Momentenmatrix zweiter Ordnung am Punkt p. Ausgeschrieben ergibt sich die Lösung zu

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \sum_i f_x(w_i)^2 & \sum_i f_x(w_i) f_y(w_i) \\ \sum_i f_y(w_i) f_x(w_i) & \sum_i f_y(w_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_x(w_i) f_t(w_i) \\ -\sum_i f_y(w_i) f_t(w_i) \end{bmatrix} \tag{4.10}$$

Um robuster gegen kleine Störungen und Rauschen zu werden, wird in der Praxis meist eine gaußsche Nachbarschaftsfunktion verwendet, die dem zentralen Pixel p das größte Gewicht zuordnet.

4.3. Mögliche Erweiterungen

Der Detektor ist in seiner jetzigen Implementierung nur bedingt für den Echtzeit-Einsatz, z.B. auf einem Roboter, geeignet. Der Grund dafür liegt in der Art und Weise wie der Skalenraum erzeugt wird. Wie in Kapitel 4 erläutert, entsteht durch die temporale Filterung der Videobilder eine Verzögerung, weil der Gaußfilter, wie in Abbildung 4.5 links zu sehen, vom Zeitpunkt der Detektion $t = 0$ aus gesehen sowohl Daten aus der Vergangenheit $t < 0$ als auch Daten aus der relativen Zukunft $t > 0$ benötigt.

Kausalitätswahrender Skalenraum

In seinem Paper „Generalized Gaussian Scale-Space Axiomatics Comprising Linear Scale-Space, Affine Scale-Space and Spatio-Temporal Scale-Space“ [Lin10] beschreibt Tony Lindenberg das Konzept eines „kausalitätswahrenden Skalenraums“. Dieser Skalenraum wird durch rekursive Filter aufgebaut, die dem Ergebnis des Gaußfilters sehr nahe kommen. Die

Notwendigkeit eingehende Videobilder und Zwischenergebnisse zu puffern würde größtenteils entfallen. Nur für den Gradienten in zeitlicher Richtung und die *non maxima suppression* der Operatorantworten müsste weiterhin jeweils ein *frame* gepuffert werden. Der Detektor würde durch Nutzung der rekursiven Filterung nur noch zwei *frames* gegenüber dem aktuellen Videobild zurück liegen. Bei den üblichen 25 Bildern pro Sekunde ergäbe sich eine Verzögerung von 80 Millisekunden, was für den Einsatz in „Echtzeit“ Szenarien ausreichend schnell wäre.

Kapitel 5

Evaluation des offline Aktionsklassifikators

In diesem Kapitel soll die Leistungsfähigkeit des Aktionsklassifikators nach Laptev evaluiert werden. Des Weiteren werden die Ergebnisse beider Detektorimplementierungen verglichen. Da die auf den erkannten *interest points* berechneten Deskriptoren auf Grund eines fehlenden Qualitätsmaßes schlecht miteinander verglichen werden können, muss bei der Validierung der beiden Implementationen auf die Klassifikationsergebnisse der gesamten Pipeline zurückgegriffen werden.

5.1. Aktionsdatenbanken

Zur Evaluation des Aktionsklassifikators dient die KTH Aktionsdatenbank, die im Folgenden vorgestellt werden soll.

KTH Videodatenbank

Die KTH[5] Datenbank [KTH] zeigt 25 Akteure, die 6 unterschiedliche Aktionen in 4 verschiedenen Szenarien ausführen. Die Datenbank besteht aus insgesamt 600 Videos in einer Auflösung von 160x120. Alle Sequenzen wurden mit einer Framerate von 25fps mit einer statischen Kamera aufgenommen. Ausschnitte aus den einzelnen Aktionsvideos sind Abbildung 5.1 zu sehen.

[5] Kungliga Tekniska högskolan(KTH), technische Universität in Schweden.

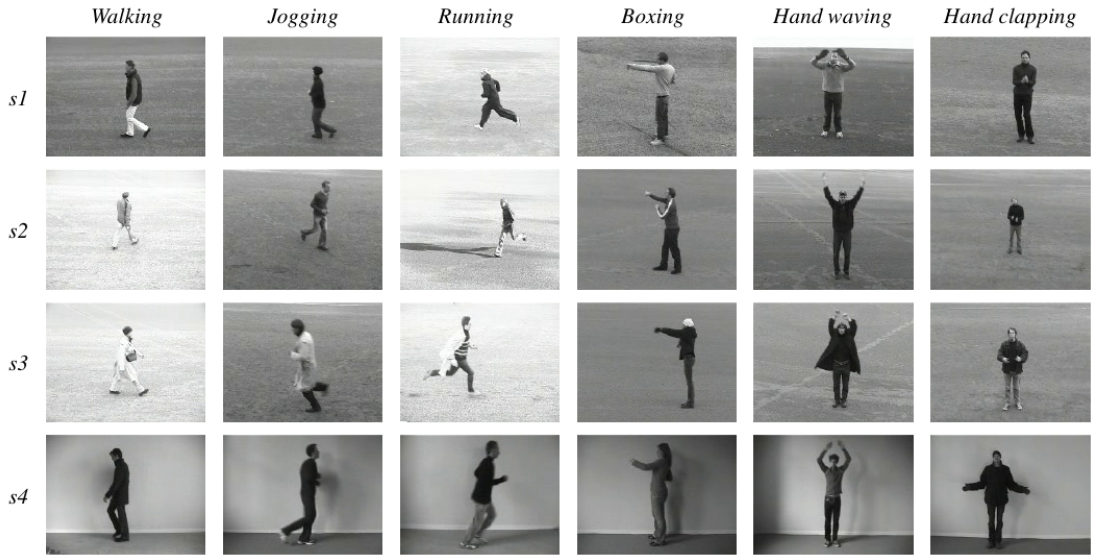


Abbildung 5.1.: KTH Human Action Database. 6 verschiedene Aktionen (gehen, joggen, rennen, boxen, winken, klatschen) in 4 unterschiedlichen Szenarien. Normal (s1), Skalenänderungen (s2), andere Kleidung (s3) und im Haus (s4). Quelle: KTH [KTH]

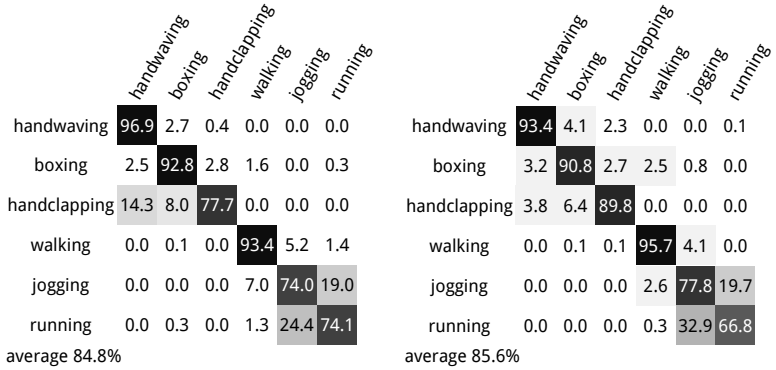
5.2. Gegenüberstellung der Detektor-Implementationen

Zur Validierung der Korrektheit des in dieser Arbeit implementierten Detektors soll der KTH Datensatz herangezogen werden. Zunächst wurden alle 600 Videos des KTH Datensatzes von beiden Detektoren bearbeitet. Hierbei wurden die Standardparameter aus Tabelle 4.1 belassen. Die Implementation von Laptev detektierte insgesamt 473.771 Punkte. Der in dieser Arbeit implementierte Detektor erkannte 475.465 Punkte. Die Differenz von 1.694 Punkten ist marginal und darauf zurückzuführen, dass einige Details des Detektors experimentell ermittelt werden mussten. Auf Grund der durchgehend guten Ergebnisse der Klassifikation wurde von einer weitergehenden Angleichung der Detektoren abgesehen.

Testaufbau

Zur Gegenüberstellung der Detektorimplementationen wurde das Klassifikationssystem so aufgebaut, wie es in Kapitel 4 beschrieben ist. Der gesamte Datensatz wurde vorab von beiden Detektoren bearbeitet. Die detektierten Punkte und die darauf extrahierten HoG und HoF Deskriptoren wurden in jeweils einer Datei für jeden Detektor gespeichert.

Die Anzahl der Cluster für das *bag of visual words* Modell wurde auf 4000 festgelegt. Als SVM kam die Implementation von *opencv* zum Einsatz. Die SVM wurde auf einen



Laptev Eigenimplementation

Tabelle 5.1.: Links: Klassifikationsergebnis mit den detektierten Punkten der Implementation von Laptev. Rechts: Klassifikationsergebnis mit den detektierten Punkten des in dieser Arbeit implementierten Detektors. Die *class confusion matrix* zeigt die durchschnittlichen Ergebnisse von 15 Durchläufen mit zufälliger Aufteilung des Test- und Trainingsdatensatzes.

RBF[6] Kern mit $\gamma = 21$ konfiguriert. Die SVM Parameter wurden vorab experimentell optimiert.

Der KTH Datensatz wurde 15 mal in jeweils etwa hälftige Aufteilungen von Trainings- und Testdaten unterteilt. Dabei wurde darauf geachtet, dass auch die Verteilung der einzelnen Aktionen zwischen Trainings- und Testdatensatz ebenfalls jeweils hälftig war. Für jede dieser 15 Unterteilungen wurde nun für beide Detektoren ein Klassifikationslauf durchgeführt.

Ergebnisse

Das gemittelte Ergebnis dieser 15 Läufe ist in Abbildung 5.1 als *class confusion matrix* zu sehen. Die Ergebnisse liegen mit 84,8% zu 85,6% sehr nah beieinander. Die geringen Abweichungen sind damit zu erklären, dass einige Details über die genaue Funktionsweise des Detektors experimentell ermittelt werden mussten. Vorallem die genaue Ausgestaltung der Deskriptorenberechnung war in [Lap05, LCSL07, LMSR08] nur sehr grundsätzlich beschrieben.

Um genauer zu analysieren, ob die Abweichungen der Ergebnisse mit der Berechnung der Deskriptoren zusammenhängen, wurde das Experiment noch einmal ausgeführt. Diesmal jedoch getrennt für den HoG und den HoF Deskriptor. Die Ergebnisse sind in Tabelle 5.2 aufgeführt. Die Unterschiede zwischen den Klassifikationsergebnissen der beiden

[6] *radial basis function.*

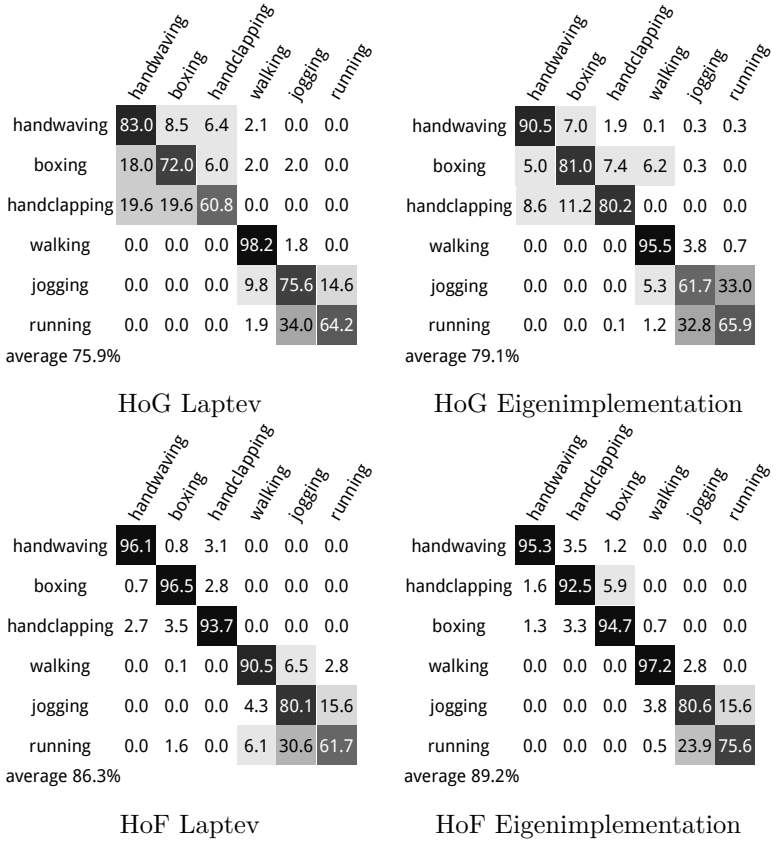


Tabelle 5.2.: Klassifikationsergebnis unter ausschließlicher Nutzung der HoG/HoF Deskriptoren auf dem KTH Datensatz

Deskriptor-Implementationen sind nicht so ausgeprägt, dass von einer grundsätzlich abweichenden Berechnung der Deskriptoren ausgegangen werden kann. Die Abweichung ist vermutlich darauf zurückzuführen, dass der hier implementierte Detektor während der Entwicklung ausschließlich mit dem KTH Datensatz getestet wurde. Die experimentell ermittelten Parameter des Verfahrens sind also speziell auf diesen Datensatz hin optimiert.

Was bei diesen Experimenten jedoch auffällt, ist der Unterschied im Klassifikationsergebnis zwischen den beiden Deskriptoren. Der HoF Deskriptor erzielt eine um etwa 10% höhere *average precision* als der HoG Deskriptor. Wie in Abschnitt 4.2 auf Seite 31 bereits erwähnt, berücksichtigt der HoG Deskriptor eher die spatialen Eigenschaften der interessanten Videoregionen, während der HoF Deskriptor vielmehr den zeitlichen Verlauf der Bildstrukturen kodiert. Der KTH Datensatz zeigt die aufgezeichneten Bewegungen vor sehr homogenen und unbewegten Hintergründen. Der HoG Deskriptor, der vor allem die räumlichen Informationen kodiert, kann damit weniger zum Klassifikationsergebnis beitragen. Der optische Fluss des HoF Deskriptors bietet mehr Anhaltspunkte für die Unterscheidung

Anzahl Cluster	4000
SVM Kernel	Radial Basis Funktion ($\gamma = 21$)
Anzahl Klassifikationsläufe	15

Tabelle 5.3.: Experimentaufbau zur Gegenüberstellung der Detektor-Implementationen

der Aktionen, da dieser hauptsächlich die eigentlichen Bewegungen kodiert. Da der KTH Datensatz mit feststehenden Kameras aufgezeichnet wurde, gibt es keinen optischen Fluss durch Eigenbewegungen der Kamera.

Es ist zu erwarten, dass die Informationen des HoG Deskriptors wichtiger werden, wenn der räumliche Kontext der Aktionen einbezogen werden kann. Die Aktion „kochen“ ist z.B. wahrscheinlicher, wenn sie in der Küche stattfindet.

Kapitel 6

Entwicklung eines online Aktionsklassifikators

Das Ziel dieser Arbeit ist die *online* Klassifikation menschlicher Aktionen. Im Gegensatz zu dem in Kapitel 4 umgesetzten Aktionsklassifikator soll nun ein System entworfen werden, dass im laufenden Betrieb weiteres Wissen über die zu klassifizierenden Bewegungen integrieren kann. Das neue System soll dabei weiterhin auf der Detektion von *space-time interest points* aufbauen. Zur Repräsentation und Klassifikation dieser *interest points* sollen in diesem Kapitel zwei neue Ansätze entwickelt werden. Wie in Abbildung 6.1 illustriert, werden somit die Stufen zwei und drei des Aktionsklassifikators durch Lösungen zur *online* Klassifikation ersetzt.

6.1. Analyse

Die Motivation für die Entwicklung eines neuen Ansatzes zur Aktionsklassifikation ergibt sich aus den Schwächen des in Kapitel 3 vorgestellten Klassifikationssystems.

- Das gesamte Training des Klassifikators erfolgt in einem Durchlauf (*batch learning*). Dafür müssen alle Trainingsbeispiele explizit gespeichert werden. Insbesondere bei mobilen Systemen wie z.B. Robotern stehen die dazu benötigten Speicherkapazitäten meist nicht zur Verfügung.
- Der zum Aufbau des Vokabulars im *bag of visual words* Modell verwendete *clustering* Algorithmus k-Means erfordert, die Anzahl der Cluster k im Vorhinein festzulegen. Ohne Vorwissen über die zu klassifizierenden Aktionen lässt sich die optimale Anzahl der Cluster jedoch nicht schätzen.

- Nachtrainieren oder Erweitern der SVM ist im laufenden Betrieb nicht möglich. Um das System zu erweitern, muss das gesamte Trainingsverfahren neu durchlaufen werden. Die Trainingsbeispiele müssen für eventuell späteres Nachtrainieren aufbewahrt werden.

Diese Einschränkungen schließen die Aktionsklassifikation von vielen Einsatzgebieten aus, die zunehmend wichtiger werden. Im Einleitungskapitel wurden bereits die sozialen Netzwerke mit ihren großen Datenmengen genannt. Diese Datenmengen lassen sich nicht mehr als Ganzes verarbeiten. Das Klassifikationssystem muss inkrementell vorgehen und dabei immer einen Datenpunkt nach dem anderen betrachten, ohne jedoch ein Datum mehrmals betrachten zu müssen. Mobile Serviceroboter, die beispielsweise Patienten mit beginnender Demenz unterstützen sollen, müssen sich mit der Zeit an schwindende Fähigkeiten ihrer Patienten anpassen. Solche Systeme sollen möglichst autonom funktionieren, der Anpassungsprozess sollte im laufenden Betrieb stattfinden.

6.2. Anforderungen an das Klassifikationssystem

Für das zu entwickelnde System ergeben sich mehrere zentrale Forderungen:

- **Inkrementelles Lernen**. Es soll jeweils nur ein Datenpunkt nach dem anderen verarbeitet werden. Explizites Wissen über vorhergehende oder nachfolgende Datenpunkte soll nicht benötigt werden. Trainingsdaten sollen nicht explizit gespeichert werden. Sie sollen implizit, durch Adaption des internen Modells spätere Prädiktionen beeinflussen.

- **Adaptives Lernen**. Moderaten Änderungen in der Aktionsausführung und den allgemeinen Umweltbedingungen soll sich das System im Laufe der Zeit anpassen können.

- **Erweiterbarkeit**. Das System soll im laufenden Betrieb durch neue Trainingsbeispiele erweitert werden können.

Durch das inkrementelle Lernen soll das System in die Lage versetzt werden, mit großen Datenmengen umgehen zu können. Der Einsatz in sich mit der Zeit verändernden Umgebungen wird durch die Forderung nach der Adaptivität des Systems berücksichtigt. Wenn die Adaptivität alleine nicht mehr ausreicht, sondern z.B. vollkommen neue Aktionsklassen gelernt werden sollen, muss das System mit neuen Trainingsbeispielen erweiterbar sein.

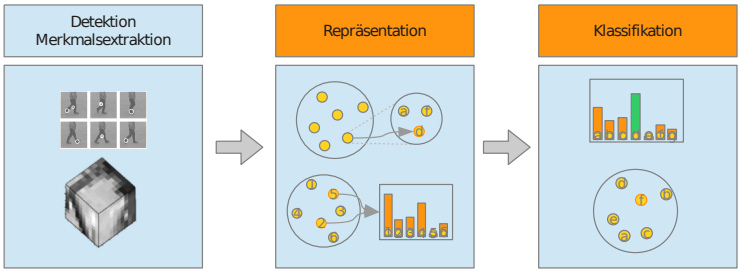


Abbildung 6.1.: Klassifikationspipeline. Die Detektion der *interest points* und die Merkmalsextraktion bleiben aus dem Verfahren nach Laptev erhalten. Beide hier entwickelten Ansätze bieten jeweils neue Lösungen zur Repräsentation und Klassifikation.

6.3. Vorüberlegungen

Um die gestellten Anforderungen in dem zu entwickelnden Klassifikationssystem umsetzen zu können, sollen zunächst einige zentrale Aspekte beleuchtet werden.

Das zu entwickelnde System soll eingehende Daten als kontinuierlichen Strom von Datenpunkten sehen, jeweils ein Datum nach dem anderen. Nachdem das Datum verarbeitet wurde, soll es nicht länger benötigt werden. Es muss also ein Weg gefunden werden, den Informationsgehalt der einzelnen Datenpunkte inkrementell in das interne Modell zu integrieren.

Inkrementelle Aktionsklassifikation

Dabei zeigen sich einige Herausforderungen, die bei dem Verfahren von Laptev (Kapitel 3) noch keine Rolle gespielt haben:

- Die Verteilung der zu lernenden Daten ist a-priori nicht bekannt, Parameter wie z.B. die Anzahl der Cluster müssen iterativ aus den Anforderungen der Daten entwickelt werden.
- Die Verteilung der zu lernenden Daten ist nicht unbedingt stationär. Die Umstände, unter denen das System eingesetzt wird, können sich ändern und das System muss sich den geänderten Umständen anpassen.
- Dem System ist, im Gegensatz zum bisherigen *batch learning*, nicht bekannt, wie viele Daten insgesamt zu bearbeiten sind.
- Annotierte Trainingsbeispiele können sich mit zu klassifizierenden Beispielen in unvorhersehbarer Weise abwechseln. Das System muss zu jeder Zeit in der Lage sein, neues Wissen zu integrieren.

Adaptivität gegenüber Störungen

Da die Einsatzbedingungen des Systems nicht ausschließlich stationär sind, muss das System aus den eingehenden Daten erkennen können, ob sich deren Eigenschaften ändern und das interne Modell gegebenenfalls adaptieren. Für das Design des Systems sind dabei folgende Formen von Störungen besonders relevant:

- **Varianz der Störsignale**. Bereits bekannte Aktionen können vor unterschiedlichen Hintergründen ausgeführt werden. Die Aktionsausführung kann durch Vordergrundobjekte überlagert werden.
- **Intra-Klassenvarianz**. Bereits bekannte Aktionen können von unterschiedlichen Akteuren verschieden ausgeführt werden.
- **Auftreten neuer Aktionsklassen**. Das System kann mit neuen Aktionsklassen konfrontiert werden. Deren Eigenschaften sollen gelernt und bei Verfügbarkeit eines annotierten Beispiels mit dem passenden Klassenlabel versehen werden.

Vergleichbarkeit von Aktionssequenzen

Die zu analysierenden Aktionen bestehen aus einer unterschiedlichen Anzahl von Aktionsprimitiven. Diese Aktionsprimitive sollen durch die Art der *interest point* Detektion und die Extraktion der HOG und HOF Deskriptoren möglichst gut erfasst werden. Die zu repräsentierenden Aktionen sind unterschiedlich lang. Auch der Detektor liefert, abhängig von der Dynamik in der aufgenommenen Szene, unterschiedlich viele Detektionen. Es müssen also unterschiedlich lange Sequenzen von an den *interest points* extrahierten Deskriptoren miteinander verglichen werden können. Die hier entwickelten Ansätze setzen auf verschiedene Ausprägungen des *bag* Modells, um diese Sequenzen vergleichbar zu machen.

Clustering

Inspiriert von der, in den letzten Jahren wachsenden, Beliebtheit verschiedener Varianten von *growing neural gas* (GNG) bei *online* Klassifikationsaufgaben, sollen sich die beiden hier entwickelten Ansätze ebenfalls darauf stützen. Beide Ansätze gruppieren die eingehenden Deskriptoren zunächst nach Ähnlichkeit. Hierfür wird GNG als *clustering* Algorithmus eingesetzt. Die genaue Funktionsweise von GNG wird im nächsten Abschnitt dargelegt. Für die Vorbetrachtung sind zunächst nur die grundlegenden Eigenschaften relevant. GNG findet eine variable Anzahl von Clustern. Die Zentren dieser Cluster folgen mit der Zeit der Verteilung der eingehenden Daten. Die Zentren können sich also im Eingaberaum bewegen. Dies stellt die Klassifikation vor große Herausforderungen, da die einzelnen Cluster durchaus auch in Regionen des Eingaberaumes wandern können, die mit einer ganz anderen Bedeutungen assoziiert sind.

Repräsentation

Wenn nun auf Basis von GNG ein *bag* Modell wie in Abschnitt 3.3 aufgebaut werden soll, kann sich die Größe des Vokabulars mit der Anzahl der gefundenen Cluster jederzeit ändern. Auch die Bedeutung einzelner „Wörter“ kann sich mit der Zeit ändern, wenn sich die ihnen zu Grunde liegenden Cluster im Eingaberaum bewegen. Das zu entwickelnde Modell muss mit diesen Dynamiken umgehen können und sich entsprechend anpassen.

Klassifikation

Für die Klassifikation der *bag* Repräsentationen wurde der DYNG Algorithmus[BC12] ausgewählt. Überlegungen, die dynamischen Repräsentationen mit Verfahren wie *edit*-Distanzen und *dynamic time warping* vergleichbar zu machen, wurden verworfen, da es keine tragfähigen Ansätze gab, die sich ändernden Bedeutungen einzelner Teile der Repräsentationsvektoren zu handhaben. Diese Aufgabe kommt in beiden Ansätzen dem DYNG Algorithmus zu.

6.4. Klassifikation mit topologischen Karten

Die in dieser Arbeit entwickelten Verfahren zur Klassifikation menschlicher Aktionen basieren auf zwei zentralen Algorithmen zum *clustering* und zur Klassifikation, die im Folgenden vorgestellt werden sollen.

6.4.1. GNG – Growing Neural Gas

Growing neural gas ist ein 1995 von Bernd Fritzke [Fri95] vorgeschlagener inkrementeller Algorithmus, der selbstorganisiert die Topologie einer Eingabeverteilung lernt. GNG wurde bereits erfolgreich zur Vektorquantisierung, zum Clustering und bei Interpolationsaufgaben angewandt. Fritzke kombiniert dabei Ideen des *neural gas* (NG) Algorithmus von Martinetz und Schulten[MS91] mit der kompetitiven Lernregel von Donald Hebb (CHL[7]). Im Gegensatz zu NG muss die Anzahl der zu findenden Cluster nicht vorab festgelegt werden, sie ergibt sich aus der Topologie des Eingaberaumes.

Algorithmus

Der Algorithmus arbeitet inkrementell, er sieht die Eingabeverteilung als kontinuierlichen Strom von Eingabevektoren. die nacheinander bearbeitet und danach nicht weiter benötigt werden. GNG ist in der Lage, sich ändernden Verteilungen der Eingabedaten zu folgen und

[7] competitive hebbian learning.

Algorithmus 1 *growing neural gas*

0. Starte mit zwei Neuronen a und b mit zufälligen Positionen w_a und w_b im $\mathbb{R}^n$.

1. Betrachte das nächste Eingabesignal x.

2. Finde die beiden dem Eingabesignal am nächsten gelegenen Neuronen s_1 und s_2.

3. Inkrementiere das Alter aller ausgehenden Kanten von s_1.

4. Addiere den quadratischen Abstand zwischen dem Eingangssignal x und dem am nächsten gelegenen Neuron s_1 zum lokalen Fehler von s_1:

$$\Delta error(s_1) = \|w_{s_1} - x\|^2$$

5. Bewege s_1 und seine direkten topologischen Nachbarn in Richtung des Eingabesignals x gewichtet durch die Faktoren ϵ_b und ϵ_n:

$$\Delta w_{s_1} = \epsilon_b(x - w_{s_1})$$
$$\Delta w_n = \epsilon_n(x - w_n) \text{ für alle topologischen Nachbarn } n \text{ von } s_1$$

6. Wenn s_1 und s_2 durch eine direkte Kante miteinander verbunden sind, setzte das Alter dieser Kante auf 0. Wenn bisher keine Kante existiert, erzeuge sie.

7. Entferne Kanten mit einem Alter größer als a_{max}. Wenn die letzte ausgehende Kante eines Neurons entfernt wurde, entferne auch das Neuron.

8. Wenn die Anzahl bisher gesehener Eingangssignale ein ganzzahliges Vielfaches des Parameters λ ist, füge eine neues Neuron ein:
   - Finde das Neuron q mit dem höchsten lokalen Fehler.
   - Finde unter den Nachbarn von q das Neuron f mit dem höchsten lokalen Fehler.
     - Füge ein neues Neuron r in der Mitte zwischen q und f ein.
       $$w_r = 0.5(w_q + w_f).$$
     - Füge zwei neue Kanten von r zu q und f ein. Entferne die direkte Kante zwischen q und f.
     - Verringere den lokalen Fehler von q und f durch Multiplikation mit einer Konstante α. Setze den lokalen Fehler von r auf den neuen Fehler von q.

9. Verringere die lokalen Fehler aller Neuronen durch Multiplikation mit einer Konstanten d.

10. Wenn weitere Eingabesignale verfügbar, gehe zu Schritt 1.

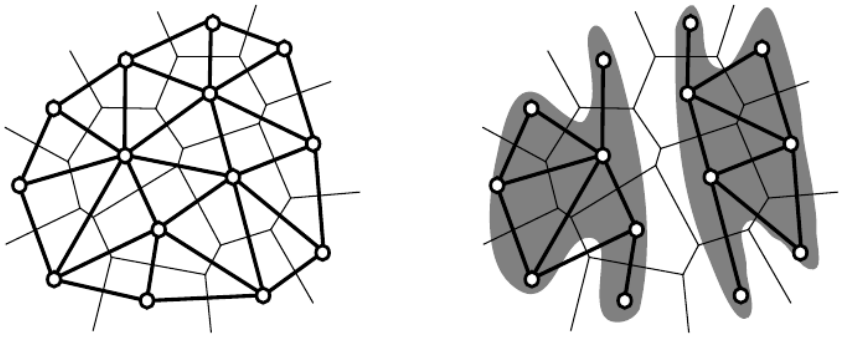


Abbildung 6.2.: Zwei Möglichkeiten die Nähe einer Menge von Punkten zu definieren. Links: Voronoi Zellen (dünne Linien) und Delaunay Triangulation (dicke Linien). Rechts: Die „induzierte“ Delaunay Triangulation wie sie vom *growing neural gas* Algorithmus erzeugt wird. Quelle [Fri95]

das *clustering* Ergebnis fortlaufend zu optimieren. Zu einer Eingabeverteilung in $\mathbb{R}^n$ wird mit jedem eingehenden Datenvektor ein verbundenes Netzwerk von Neuronen erzeugt, in dem jedes Neuron N eine Position w_N im $\mathbb{R}^n$ hat. Die Positionen der einzelnen Neuronen im Eingaberaum dienen als Referenzvektoren für die Cluster, die sie repräsentieren.

Der Algorithmus startet mit zwei Neuronen mit jeweils zufällig initialisierten Positionen im $\mathbb{R}^n$ und baut die Topologie des Netzwerkes sukzessive aus den Eingabesignalen auf. Das Grundprinzip ist dabei eine Variante der Hebb'schen Lernregel („what fires together, wires together“).

> Für jedes Eingangsignal x werden die beiden nach euklid'scher Norm am nächsten gelegenen Neuronen mit einer Kante verbunden.

Der durch die Anwendung dieser Regel entstehende Graph ist die sogenannte „induzierte“ Delaunay Triangulation (Abbildung 6.2). Aus der Hebb'schen Lernregel ergibt sich, dass Neuronen, die nicht auf der gemeinsamen Mannigfaltigkeit der Eingabedaten liegen, keine Kanten entwickeln, da sie nie zu den beiden am nächsten zu einem Eingabesignal gelegenen Neuronen gehören. Diese ungenutzten Neuronen können mit der Zeit entfernt werden. Zu diesem Zweck hat jede Kante ein Alter. Ist das Alter einer Kante größer als das festgelegte Höchstalter a_{max}, wird die Kante entfernt. Wenn dadurch ein Neuron keine Kanten zu anderen Neuronen mehr haben sollte, wird es ebenfalls entfernt. Um möglichst alle Neuronen optimal zu nutzen, müssen sie in Regionen platziert werden, in denen Eingabedaten zu erwarten sind. Der Algorithmus versucht dies zu erreichen, indem für jedes Eingabesignal x die durch Kanten verbundenen Nachbarneuronen in Richtung des Vektors x bewegt werden.

Abbildung 6.3 zeigt ein GNG Netz, das durch eine spiralförmige Verteilung von Eingabesignalen entwickelt wurde. Die Eingabesignale wurden zufällig und gleichverteilt über

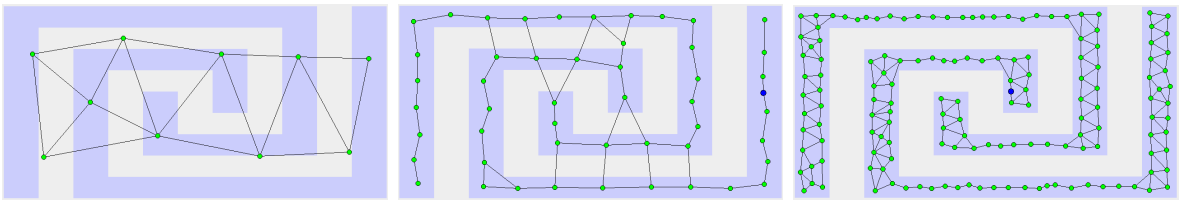


Abbildung 6.3.: ***growing neural gas*** **folgt einer spiralförmigen Verteilung von Eingabesignalen. Von Links nach Rechts: 10, 50, 150 Neuronen. Der Algorithmus hat jeweils so viele Iterationen durchlaufen, bis das erzeugte Netz die angegebene Zahl an Neuronen umfasste. Diese Illustration wurde mit dem DemoGNG Applet http://www.demogng.de/ erstellt**

den bläulich markierten Regionen erzeugt. Es zeigt sich, dass GNG in der Lage ist, auch schwierige Verteilungen abzubilden.

Erweiterungen

Für den in Abschnitt 7 entwickelten Ansatz wurde GNG so erweitert, dass jedes Neuron eine eindeutige ID aus der Menge {1, ..., I} besitzt, wobei I die maximale Anzahl der Neuronen im Netz ist. Dies erlaubt es, beliebig dimensionale Vektoren in eine definierte Menge natürlicher Zahlen abzubilden. IDs von Neuronen, die aus dem Netz entfernt wurden, können erneut vergeben werden.

6.4.2. DYNG – Dynamic Online Growing Neural Gas

DYNG ist ein von Beyer und Cimiano in [BC12] entwickelter *online* Algorithmus zur Klassifikation von Datenströmen. DYNG basiert auf OGNG[8] [BC11], einem inkrementellen Klassifikator, der auf den Grundideen von GNG aufsetzt und diese um *labelling*- und Prädiktionsstrategien erweitert.

Der DYNG Algorithmus baut zunächst ähnlich zu GNG ein Netzwerk verbundener Neuronen auf, dass die Topologie der Eingabedaten widerspiegelt. Um dieses Netzwerk zur Klassifikation nutzen zu können, muss es um zwei Funktionen erweitert werden. Eine *labelling* Funktion, die den einzelnen Neuronen Labels zuweist. Darüber hinaus braucht es eine *prediction* Funktion, die Labels zu bisher ungesehenen Beispielen zuweist. Als *labelling* Funktion verwendet DYNG dabei die *relabel* Strategie des OGNG. Als *prediction* Funktion wird die *single linkage* Strategie des OGNG eingesetzt.

Im Gegensatz zu OGNG, nutzt DYNG Labelinformationen nicht nur zur Klassifikation, sondern auch um die Steuerung des Netzwerkwachstums an den Anforderungen der Daten

[8] *online growing neural gas.*

auszurichten. Das Wachstum des Netzes wird beschleunigt, wenn bisher ungesehene Labels eingehen und es wird gedämpft wenn der Klassifikationsfehler konvergiert.

In dieser Arbeit wird die Referenzimplementation des DYNG Algorithmus von O. Beyer verwendet.

Parameter

Die Parameter des GNG und DYNG Algorithmus sind in der nachfolgenden Tabelle mit ihren jeweiligen Bedeutungen aufgeführt.

Parameter	Bedeutung
λ	Bestimmt nach wie vielen gesehenen Datenpunkten ein neues Neuron eingefügt wird.
ϵ_b	Faktor mit dem das am nächsten gelegenen Neurons in Richtung des Eingabesignals bewegt wird.
ϵ_n	Faktor mit dem die topologischen Nachbarn des am nächsten gelegenen Neurons in Richtung des Eingabesignals bewegt werden.
d	Faktor mit der die lokalen Fehler der Neuronen reduziert werden.
α	Faktor mit der die lokalen Fehler der beiden zu dem gegebenen Eingangssignal nächst gelegenen Neuronen reduziert wird.
a_{max}	Maximales Alter, nach dem eine Kante gelöscht wird.

Tabelle 6.1.: Parameter des *growing neural gas* Algorithmus

Kapitel 7

Sequenzieller Ansatz

Dieser Ansatz orientiert sich, ausgehend von den guten Ergebnissen in *offline* Szenarien, an der Vorgehensweise von Laptev (Kapitel 3). Der Aufbau ist sequenziell, alle eingehenden *interest points*, bzw. die auf ihnen berechneten Deskriptoren, durchlaufen die selbe Pipeline. Der grundsätzliche Ablauf ist auch bei diesem Verfahren wieder dreischichtig:

- Detektion der *interest points* mit dem Harris3D Detektor.
- Repräsentation als *adaptive bag of visual words* mit variablem Vokabular.
- Klassifikation der *adaptive bag of visual words* Histogramme mittels DYNG.

Die Detektion der *interest points* und die Berechnung der HOG und HOF Deskriptoren (Abschnitt 4.2 auf Seite 31) ist identisch zu dem Verfahren von Laptev. Große Anpassungen sind jedoch bei der Repräsentation und Klassifikation notwendig. Die Repräsentation der detektierten Punkte als *bag of visual words* ist, wie sich im nächsten Abschnitt zeigen wird, bei der *online* Klassifikation nicht ohne Weiteres möglich.

Bei Laptevs Verfahren (Abschnitt 3.3 auf Seite 22) wurde das Modell aufgebaut, indem zuerst die Trainingsdaten mit k-Means in Cluster ähnlicher Deskriptoren eingeteilt wurden. Diese Cluster dienten als Vokabular für das *bag of visual words* Modell, wobei die ID jedes einzelnen Clusters für ein Wort des Vokabulars stand. Die Deskriptoren einer zu klassifizierenden Videosequenz wurden nach euklidischem Abstand dem jeweils nächsten Wort des Vokabulars zugeordnet und in das Frequenzhistogramm aller Wörter der Sequenz aufgetragen. Da die eingehenden Daten als kontinuierlicher Strom von Datenpunkten aufgefasst werden sollen, lassen sich vorab keine Annahmen über die optimale Anzahl an Clustern und deren Position im Eingaberaum treffen. Der k-Means Algorithmus lässt sich unter diesen Umständen nicht mehr einsetzen, da er die Trainingsdaten als Ganzes benötigt. Wenn a-priori nichts über die Eingabedaten bekannt ist, müssen die Cluster inkrementell aus den Daten ermittelt werden. Dieser Ansatz verwendet dafür *growing neural gas*.

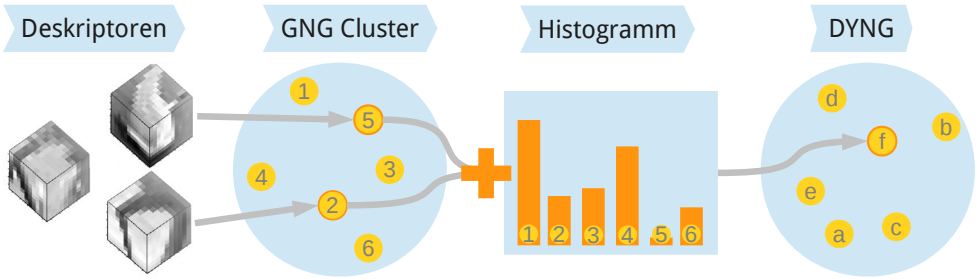


Abbildung 7.1.: Ablaufdiagramm zum sequentiellen Ansatz. Die auf den Nachbarschaften der *interest points* berechneten Deskriptoren werden zunächst in einem *clustering* Schritt gruppiert und das am nächsten gelegene Neuron des GNG Netzes wird ermittelt. Die eindeutigen IDs der jeweils nächsten Neuronen werden in ein gemeinsames Histogramm eingetragen. Anschließend wird das Histogramm nach euklidischer Norm normalisiert. Das normalisierte Histogramm dient als *feature* Vektor für die Klassifikation mittels DYNG

Das bisherige *bag of visual words* Modell ist auf stabile Clusterzentren als Repräsentanten einzelner Wörter des Vokabulars angewiesen. GNG passt jedoch die Anzahl und die Position der Cluster ständig an die Verteilung der eingehenden Daten an. Um das zu berücksichtigen, muss das Repräsentationsmodell zum *adaptive bag of visual words* Modell weiter entwickelt werden. Die SVM aus dem Verfahren von Laptev lässt sich unter diesen Umständen ebenfalls nicht einsetzen, da sie, ähnlich wie der k-Means, eine globale Sicht auf alle Trainingsdaten voraussetzt. Als Klassifikator wird in diesem Ansatz DYNG eingesetzt.

7.1. Repräsentation als adaptive Bag of Visual Words

Der sequenzielle Ansatz behält das *bag of visual words* Modell aus dem bisherigen Verfahren bei. Jedoch müssen für den Einsatz in einem *online* Setting einige Modifikationen vorgenommen werden, um den besonderen Herausforderungen gerecht zu werden. Wie zu Beginn des Kapitels bereits erwähnt, ist eine dieser Herausforderungen, dass der Aktionsklassifikator nicht von einer statischen Umweltsituation ausgehen kann. Das System soll in der Lage sein, im laufenden Betrieb neue Aktionen zu lernen und sich Variationen in der Ausführung bereits gelernter Aktionen im Laufe der Zeit anzupassen. Der bisher eingesetzte k-Means Algorithmus ist in seiner allgemeinen Form nicht in der Lage, die einmal ermittelten Clusterzentren im laufenden Betrieb an sich ändernde Verteilungen der Eingabedaten anzupassen. Für diesen Ansatz muss also eine andere Vorgehensweise gefunden werden, um das Vokabular für das *bag of visual words* Modell zu generieren. Der GNG Algorithmus eignet sich gut für diesen Zweck. Der Algorithmus optimiert die Clusterzentren inkrementell, ein einmal gesehenes Datum wird nicht weiter benötigt. Darüber hinaus

ist er in der Lage, nicht-stationären Verteilungen der Eingabedaten im Laufe der Zeit zu folgen.

Aufbau des Vokabulars

Der generelle Aufbau des *adaptive bag of visual words* Modells ist ähnlich dem Vorgehen aus Kapitel 3. Die eingehenden Deskriptoren werden durch einen *clustering* Schritt nach Ähnlichkeit zusammengefasst und durch die Nummer des Clusterzentrums ersetzt, dem sie nach euklidischen Abstand am nächsten sind. Für jedes Cluster wird nun in einem Histogramm aufgetragen, wie viele Deskriptoren ihm zugeordnet wurden. Das Vokabular des *adaptive bag of visual words* Modells wird iterativ mittels GNG erzeugt. Dadurch ergeben sich teils gravierende Änderungen gegenüber dem bisherigen Modell:

- Die Größe des Vokabulars kann sich ändern. GNG beginnt initial mit zwei Neuronen und baut das Netzwerk sukzessive bis zu der maximalen Anzahl an Neuronen aus. Neuronen, die nicht zum *clustering* Ergebnis beitragen, weil sie nicht auf der gemeinsamen Mannigfaltigkeit der Eingabedaten liegen, können vom Algorithmus entfernt werden.
- Die Bedeutung der Clusterzentren kann sich mit der Zeit ändern, da die Neuronen langsam in Richtung der Eingabevektoren gezogen werden und damit eventuell Deskriptoren repräsentieren, die für ganz andere Aktionen typisch sind.

Wenn neue Cluster dazukommen, wird das Histogramm um einen Eintrag erweitert. Fallen Cluster weg, weil Neuronen aus dem GNG Netz gelöscht werden, bleibt der entsprechende Eintrag des Histogramms leer. Dieser Umstand muss bei der Klassifikation beachtet werden. Das Histogramm wird anschließend normalisiert. In diesem Ansatz wird nicht die *city block* Norm wie in Laptev's Verfahren verwendet, sondern die euklidsche Norm.

7.2. Klassifikation mittels DYNG

Die normalisierten *adaptive bag of visual words* Histogramme der zu untersuchenden Videosequenz werden mit DYNG klassifiziert. Die Schwierigkeit bei der Klassifikation der *adaptive bag of visual words* Repräsentation liegt darin, dass sich die Dimensionalität des Histogramms jederzeit ändern kann, wenn ein Neuron aus dem *clustering* Netz entfernt wird. Um diese Histogramme dennoch direkt als *feature* Vektoren für die Klassifikation verwenden zu können, wurde DYNG so erweitert, dass sich nachträglich einzelne Koordinaten aus den Positionsvektoren der Neuronen entfernen lassen. Praktisch wurde dies so realisiert, dass die entsprechende Koordinate bei allen Neuronen im Netz auf null gesetzt wird und somit bei der euklidischen Distanzberechnung keinen Einfluss mehr hat.

Sollte sich ein neues Cluster aus den Daten herausbilden, dem eben diese ID zugewiesen wird, kann sich die entsprechende Dimension in den Positionsvektoren durch die Adaptionsmechanismen von DYNG im Laufe der Zeit wieder herausbilden und einen Beitrag zum Klassifikationsergebnis leisten.

7.3. Plastizität des Klassifikationsnetzes

Der KTH Datensatz umfasst 600 Aktionssequenzen. Jede Aktionssequenz wird durch jeweils ein *adaptive bag of visual words* Histogramm repräsentiert. Bei einer hälftigen Aufteilung in Trainings- und Testdatensatz, stehen dem DYNG Netz also nur 300 Trainingsbeispiele zur Verfügung. DYNG ist als inkrementeller Klassifikationsalgorithmus eher auf größere Datenmengen und damit längere Adaptionsprozesse ausgelegt. Die beiden Parameter ϵ_b und ϵ_n, die die Stärke der Adaption der Positionen des am nächsten gelegenen Neurons und seiner topologischen Nachbarn bestimmen, ist entsprechend niedrig angesetzt. Das führt zu einer höheren Stabilität des Gesamtnetzes. Wenn nun aber nur wenige Trainingsbeispiele zur Verfügung stehen, muss das Netz sich schneller anpassen, die langfristige Stabilität ist weniger wichtig. Dieser Umstand ist als *stability plasticity dilemma* bekannt. Um dies zu berücksichtigen wurde der Faktor μ eingeführt, der die Beweglichkeit der Neuronen in Bezug auf die für langfristige Stabilität optimierten Standardparameter $\hat{\epsilon}_b$ und $\hat{\epsilon}_n$ angibt :

$$\epsilon_b = \mu\hat{\epsilon}_b \tag{7.1}$$

$$\epsilon_n = \mu\hat{\epsilon}_n \tag{7.2}$$

Für den KTH Datensatz hat sich ein Wert von $\mu = 3$ als optimal erwiesen.

7.4. Nutzung topologischer Informationen

Während der Experimente hat sich gezeigt, dass gerade die Trennung der Fußaktionen „gehen“, „joggen“ und „rennen“ problematisch ist. Dieser Umstand soll in Abbildung 7.2 näher beleuchtet werden. Für die Abbildung wurden zunächst alle Deskriptoren mit k-Means in 400 Cluster eingeteilt. Dann wurde für jeden Deskriptor die jeweilige Clusterzugehörigkeit in ein Histogramm aufgetragen. Die einzelnen *bins* des Histogramms wurden anschließend durch die Anzahl der Deskriptoren geteilt. Die selbe Prozedur wurde auch für jede Aktionsklasse einzeln durchgeführt. Die in der Abbildung zu sehenden Diagramme stellen die absolute Differenz des Mittels aller Deskriptoren zu den jeweiligen normalisierten Histogrammen der einzelnen Aktionsklassen dar. Es ist deutlich zu sehen, dass die Handaktionen

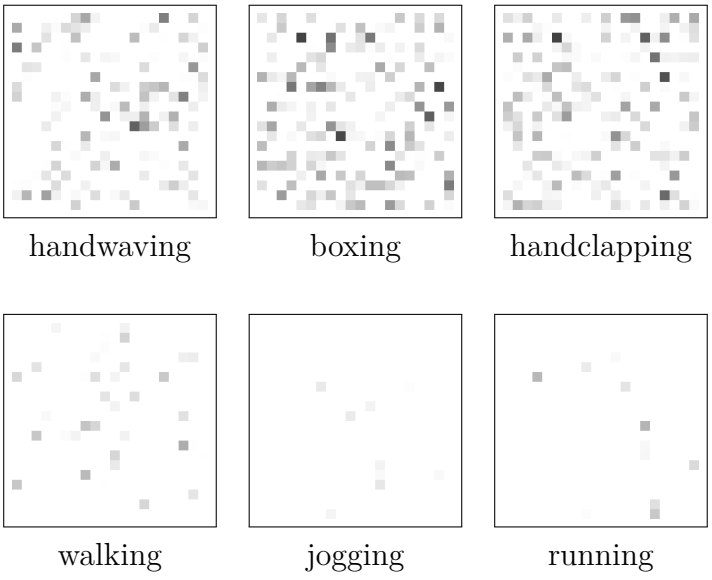


Abbildung 7.2.: Abweichung der Cluster einzelner Deskriptoren vom Mittelwert. Für diese Grafik wurden zunächst alle Deskriptoren, die auf dem KTH Datensatz berechnet wurden mit k-Means in 400 Cluster eingeteilt. Die Zugehörigkeit der einzelnen Deskriptoren zu den jeweiligen Clustern wurde in einem Histogramm aufgetragen und zu eins normiert. Für jede der 6 Aktionsklassen des Datensatzes wurde ebenfalls ein normiertes Histogramm berechnet. Die abgebildeten Graphiken zeigen nun die absolute Differenz der einzelnen Cluster der jeweiligen Aktionsklassen zum Mittel aller Aktionsklassen. Es ist zu erkennen, dass sich die Handaktionen deutlicher vom Mittel abheben als die Fußaktionen.

wesentlich stärkere Abweichungen vom Mittel aller Aktionen aufweisen, als die Fußaktionen. Je stärker aktionsspezifische Charakteristika ausgeprägt sind, desto leichter lassen sie sich voneinander unterscheiden. Die wenig charakteristischen Ausprägungen der Fußaktionen stellen eine besondere Herausforderung für die Klassifikation dar.

Viele Cluster unterscheiden sich nur gering zwischen den einzelnen Aktionen. Diese Cluster repräsentieren zum Teil Nachbarschaften von *interest points*, die auf verschiedenen Störungen, wie z.B. Kompressionsartefakten, detektiert wurden. Da die Cluster mit geringer Varianz nur wenig Information zur Unterscheidung der Aktionsklassen beitragen, wurde untersucht, ob die topologischen Informationen des GNG Netzes einen Beitrag leisten können.

Aufbau

Das trainierte GNG Netz bildet die Topologie der Eingabedaten nach. Wie in Abbildung 6.3 auf Seite 50 zu sehen, kann GNG dabei auch unverbundene Subnetze auf den isolierten Clustern der Eingabeverteilung herausbilden. Es wurde versucht diesen Umstand für die Klassifikation zu nutzen, in dem nicht mehr die IDs einzelner Neuronen in das *adaptive*

Parameter	Vorgabe	Bedeutung
μ	3	Faktor, der die Beweglichkeit der Neuronen im Klassifikationsnetz bezogen auf die Standardparameter bestimmt.

Tabelle 7.1.: Parameter des sequentiellen Ansatz und ihre Bedeutung

bag of visual words Modell eingehen, sondern IDs isolierter Subgraphen des GNG Netzes. Alle durch Kanten verbundenen Neuronen würden somit die selbe ID tragen.

Zu diesem Zweck wurde GNG so erweitert, dass bei jedem Einfügen oder Entfernen einer Kante geprüft wird, ob durch diese Operation zwei Subgraphen verschmolzen oder isoliert wurden. Wenn zwei Subnetze verschmelzen, werden allen Neuronen des kleineren Netzes mit der ID des größeren Netzes markiert. Die Größe des Netzes richtet sich hierbei nach der Anzahl der Neuronen und nicht nach dem abgedeckten Areal im Eingaberaum. Es wird generell davon ausgegangen, dass das größere Subnetz auch den größeren Teil der Eingabeverteilung abdeckt.

Das Verschmelzen oder Isolieren zweier Subnetze bedeutet eine zusätzliche Störung für die nachfolgende Klassifikation. Zuvor waren beide Netze durch ihre ID mit dem entsprechenden *bin* im Histogramm assoziiert. Nach der Verschmelzung bleibt nur die ID des zuvor größeren Netzes erhalten. Das zusammengelegte Netz repräsentiert nun aber ein größeres Areal im Eingaberaum und damit eventuell auch ein größeres Repertoire an Aktionsprimitiven. Dadurch, dass das kleinere Netz dem größeren zugeschlagen wird, soll die Störung der Klassifikation minimiert werden, die die daraus resultierenden Effekte letztendlich kompensieren muss.

Der umgekehrte Fall tritt ein, wenn ein Netz in zwei Subnetze aufgeteilt wird. Dann wird der selbe Fundus an Aktionsprimitiven durch zwei *bins* im Histogramm repräsentiert. Auch dieser Effekt muss in der Klassifikation kompensiert werden.

Die Kompensation dieser Effekte obliegt den Adaptionsmechanismen des DYNG Algorithmus. Da eine Änderung der Cluster bei diesem Ansatz nur durch gelabelte Beispiele angestoßen werden kann, liegt der Klassifikation dazu immer mindestens ein Trainingsbeispiel in der neu entstandenen Repräsentation vor.

Ergebnisse

Bei den Experimenten mit dieser Variante hat sich gezeigt, dass der Unterraum auf dem die Eingabedaten im Eingaberaum liegen, sehr gleichmäßig besetzt ist. Es bilden sich nur wenige kleinere Subnetze heraus, die nur wenig diskriminative Information bezüglich der Klassifikation aufweisen. Durch gezieltes Parametertuning konnte zwar eine gute Partitionierung des Netzes erreicht werden, die einzelnen Cluster haben sich jedoch als zu instabil und Parameterabhängig erwiesen um die Klassifikation darauf aufzubauen.

Kapitel 8

Hierarchischer Ansatz

In diesem Kapitel soll ein hierarchischer Ansatz zur *online* Aktionsklassifikation entworfen werden. Dieses Verfahren soll so ausgerichtet sein, dass das System moderaten Änderungen der Umweltbedingungen folgen kann, ohne dazu explizit neue Trainingsbeispiele zu benötigen. Um dies zu erreichen sollen Konzepte des *semi-supervised learning* genutzt werden.

Der allgemeine Ablauf ist auch bei diesem Ansatz wieder dreischichtig:

- Detektion der *interest points* mit dem Harris3D Detektor
- Repräsentation als *bag of class labels* Histogramm
- Klassifikation der Histogramme nach dem häufigsten Aktionslabel

Repräsentation und Klassifikation sind hierarchisch in zwei Ebenen organisiert. Die erste Ebene besteht aus einem *clustering* Schritt. Zu den eingehenden *feature* Vektoren werden die jeweils am nächsten gelegenen Clusterzentren ermittelt. Zu jedem dieser Cluster gibt es ein eigenes unabhängiges DYNG Netz. Diese nachgelagerten Netze bilden die zweite Ebene. Sie werden direkt mit den Klassenlabels trainiert. Bei der Klassifikation bekommt also jeder eingehende Deskriptor das Klassenlabel des gewinnenden Neurons aus dem zweiten Netz. Diese Klassenlabel werden über die gesamte zu klassifizierende Sequenz in einem *bag of class labels* Histogramm aufgetragen. Das häufigste Label wird als Ergebnis der Klassifikation ausgewählt.

8.1. Repräsentation als Bag of Class Labels

Die zu klassifizierende Szene wird auch in diesem Ansatz wieder durch ein gemeinsames Histogramm repräsentiert. Im Unterschied zu dem vorhergehenden Ansatz werden die Deskriptoren jedoch nicht nach Ähnlichkeit gruppiert in ein Histogramm eingetragen, sondern

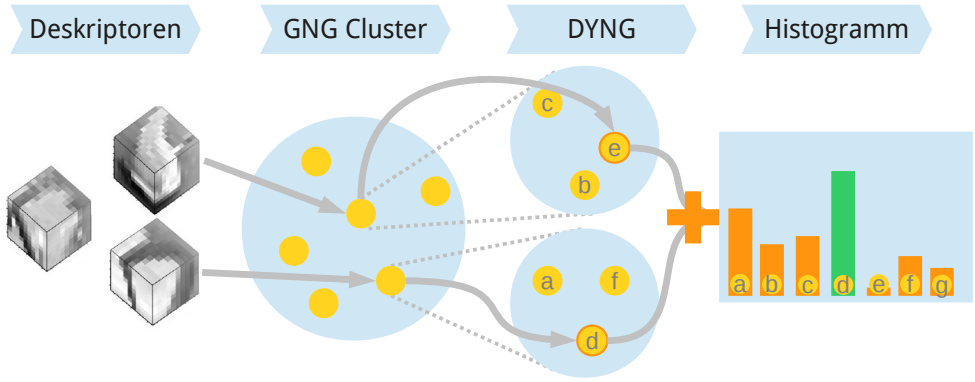


Abbildung 8.1.: Ablaufdiagramm zum hierarchischem Ansatz. Die auf den Nachbarschaften der *interest points* berechneten Deskriptoren werden zunächst in einem *clustering* Schritt gruppiert und das am nächsten gelegene Neuron des GNG Netzes wird ermittelt. Hinter jedem dieser Neuronen liegt noch ein zweites Netz, durch das der eingehende Deskriptor klassifiziert wird. Die dabei ermittelten Klassenlabel werden in ein gemeinsames Histogramm eingetragen. Die Gesamtsequenz bekommt das Label des häufigsten Eintrages im Histogramm.

die einzelnen Deskriptoren werden klassifiziert und jedes ermittelte Aktionslabel votiert für das Aktionslabel der Gesamtsequenz. Dazu wird zu jedem eingehenden Deskriptor zunächst das jeweils nächst gelegene Cluster aus dem GNG Netz der ersten Ebene ermittelt. Hinter jedem dieser Cluster liegt in der zweiten Ebene ein unabhängiges DYNG Netz, durch das der jeweilige Deskriptor klassifiziert wird. Das dabei ermittelte Aktionslabel wird in das gemeinsame *bag of class labels* Histogramm aufgetragen. Da auch bei diesem Ansatz wieder GNG verwendet wird, kann sich die Anzahl der Cluster jederzeit ändern. Sollte ein Cluster entfernt werden, wird das nachgelagerte DYNG ebenfalls entfernt. Kommt ein neues Cluster hinzu, wird das DYNG Netz eines der beiden Nachbarneuronen des neuen Neurons kopiert und dem neuen Neuron zugeordnet. Dass ein neues Neuron initial zwei Nachbarn hat, ergibt sich aus der Art und Weise wie bei GNG neue Neuronen in das Netz integriert werden (siehe Abschnitt 6.4.1). Damit wird erreicht, dass das neue Netz direkt zur Klassifikation beitragen kann.

8.2. Klassifikation der Bag of Class Labels

Die Klassifikation der *bag of class labels* erfolgt bei diesem Ansatz als *majority vote*. Die Gesamtsequenz bekommt das Label, welches die meisten Stimmen im Wahlprozess bekommen hat und damit als häufigstes im *bag of class labels* vertreten ist.

8.3. Gewichtung nach Information-Gain

Bei den bisherigen Experimenten hat sich gezeigt, dass manche Cluster in jeder Aktionsklasse nahezu gleich häufig vorkommen. Diese Cluster können z.B. mit Störungen wie Kompressionsartefakten assoziiert sein (siehe Abschnitt 7.4). Die Deskriptoren, die diesen Clustern zugeordnet werden, tragen nur wenig zum Klassifikationsergebnis bei und können dieses sogar verschlechtern, wenn sie in großer Zahl immer für die selbe Aktionsklasse votieren. Dieses Verhalten ist experimentell vor allem bei den Fußaktionen aufgefallen, bei denen der Klassifikator die Tendenz zeigte, sich für die mittlere Klasse „joggen“ zu entscheiden.

Um diesen Umstand zu berücksichtigen, sollen die einzelnen Voten nach dem zu erwartenden Beitrag zum Klassifikationsergebnis gewichtet werden. Dazu wird ein Maß für den *information gain* eines einzelnen Deskriptors benötigt.

Der bisherige Ablauf ist, dass zu jedem Deskriptor das jeweils nächst gelegene Neuron aus dem GNG Netz der ersten Schicht ermittelt wird. Das dem entsprechenden Cluster nachgelagerte DYNG Netz klassifiziert nun den Deskriptor und das dabei ermittelte Aktionslabel trägt als einzelne Stimme zum Wahlprozess des Labels der Gesamtsequenz bei, wobei jede Stimme mit dem selben Gewicht eingeht.

In den Netzen der zweiten Ebene sind somit alle Aktionslabels der Deskriptoren vertreten, die in dem Bereich des Eingaberaums liegen, der von dem entsprechenden Neuron der ersten Schicht abgedeckt wird. Bei den Deskriptoren, die mit Störungen oder bei vielen Aktionsklassen vorkommenden Bewegungsprimitiven assoziiert sind, ist also zu erwarten, dass viele verschiedene Aktionslabels in dem entsprechenden DYNG Netz zu finden sind.

Aus der Labelverteilung der DYNG Netze der zweiten Ebene lässt sich also ablesen, wie die entsprechenden Deskriptoren auf die einzelnen Aktionsklassen verteilt sind. Ist die Verteilung sehr gleichmäßig, ist ein geringer Beitrag zur sicheren Unterscheidung der Aktionsklassen zu erwarten. Die Stimme des Deskriptors kann also geringer bewertet werden. Ist in einem DYNG Netz jedoch eine Aktionsklasse signifikant öfter vertreten als die anderen, kann davon ausgegangen werden, dass die entsprechenden Deskriptoren charakteristische Primitive für diese Aktionsklasse repräsentieren. Die Stimme des Deskriptors sollte also möglichst hoch bewertet werden. Das Gewicht der Stimme berechnet sich dabei aus der Anzahl l_1 des am häufigsten vorkommenden Labels und der Anzahl l_2 des zweit häufigsten wie folgt:

$$weight = (1 - (l_2/l_1))^d. \tag{8.1}$$

Die nichtlineare Gewichtung mit einem Exponenten $d > 3$ hat sich als effektiver als eine lineare Gewichtung erwiesen. Alle Experimente mit diesem Ansatz wurden mit $d = 5$ durchgeführt.

8.4. Semi-supervised learning

Beim *semi-supervised learning* werden sowohl gelabelte als auch ungelabelte Beispiele zum Training des Klassifikators genutzt. Dabei gibt es mehrere Möglichkeiten. Für diesen Ansatz soll das sogenannte *self training* umgesetzt werden, bei dem der Klassifikator ungelabelte Beispiele zunächst klassifiziert und sich anschließend mit der Ergebnis seiner eigenen Prädiktion selbst trainiert.

Wie in Abschnitt 7.4 auf Seite 55 dargelegt, bieten insbesondere die Fußaktionen nur wenige ausgeprägte Unterscheidungsmerkmale. Der Klassifikator neigt dazu, eine der Fußaktionen bevorzugt auszuwählen. Damit diese Tendenz durch das *semi supervised learning* nicht verstärkt wird, sollen nur solche Beispiele zum weiteren Training ausgewählt werden, bei dem die Prädiktion mit hoher Wahrscheinlichkeit richtig war. Um die Wahrscheinlichkeit direkt zu schätzen müsste das System sein Klassifikationsergebnis in regelmäßigen Abständen anhand von gelabelten Beispielen überprüfen können. Da das Klassifikationssystem jedoch nicht davon ausgehen kann, in regelmäßigen Abständen annotierte Beispiele zur Verfügung zu haben, muss ein anderes Maß für die Sicherheit der Prädiktion gefunden werden.

Sicherheit der Prädiktion

Diese Maß kann aus dem Wahlprozess bei der Klassifikation abgeleitet werden. Wie in Abschnitt 8.1 bereits ausgeführt, trägt jeder aus der zu klassifizierenden Sequenz extrahierter Deskriptor mit einer Stimme zum Gesamtergebnis der Klassifikation bei. Letztendlich ergibt sich im *bag of class labels* also eine „Wahlurne“ mit allen Einzelstimmen. Die Verteilung der Klassenlabel soll nun zur Berechnung der Prädiktionssicherheit für das ausgewählte Label l herangezogen werden. Dabei wird das Verhältnis der Anzahl der Voten für das ausgewählte Label $votes_l$ zu der Anzahl der Gesamtstimmen $votes$ gebildet.

$$c(l) = \frac{votes_l}{votes} \tag{8.2}$$

Auswahl der Trainingsbeispiele

Zur Auswahl der Trainingsbeispiele wurde für jede Aktionsklasse die Prädiktionssicherheit der letzten 200 Prädiktionen gespeichert. Wenn die Prädiktionssicherheit des vorliegenden Beispiels nun das 1.4 Fache der durchschnittlichen Sicherheit seiner Klasse übertrifft, wird das Beispiel als neues Trainingsbeispiel ausgewählt.

Kapitel 9

Evaluation

In diesem Kapitel soll die Leistungsfähigkeit der beiden in dieser Arbeit entwickelten Ansätze zur *online* Aktionsklassifikation evaluiert und diskutiert werden.

9.1. Testaufbau

Zur Evaluation der Aktionsklassifikatoren wird der KTH Datensatz herangezogen. Zunächst wurden die 600 Videodateien vorab von dem in Kapitel 4 implementierten Detektor bearbeitet. Die detektierten *interest points* und die an diesen Punkte extrahierten Deskriptoren wurden in einer Datei gespeichert. Für jeden Ansatz wurden 15 Durchläufe mit jeweils hälftiger Aufteilung des Datensatzes in Test- und Trainingsdaten durchgeführt. Dabei wurde darauf geachtet, dass auch die Verteilung der einzelnen Aktionen zwischen Trainings- und Testdatensatz ebenfalls jeweils hälftig war. Für jede dieser 15 Unterteilungen wurde nun für beide Detektoren ein Klassifikationslauf durchgeführt, für beide Ansätze wurden dabei jeweils optimierte Parameter verwendet.

Sequenzieller Ansatz

Das GNG Netz wurde auf eine maximale Anzahl von 4000 Neuronen limitiert. Die Wachstumsrate des Netzes war $\lambda = 50$. Der Faktor, der die Plastizität des klassifizierenden Netzes bestimmt, war auf $\mu = 3$ festgelegt.

Hierarchischer Ansatz

Das GNG Netz der ersten Ebene wurde auf eine maximale Anzahl von 1500 Neuronen limitiert. Die Wachstumsrate des Netzes war $\lambda = 50$. Die maximale Anzahl an Neuronen in den DYNG Netzen der zweiten Ebene war auf 200 limitiert. Es wurde sowohl die Gewichtung

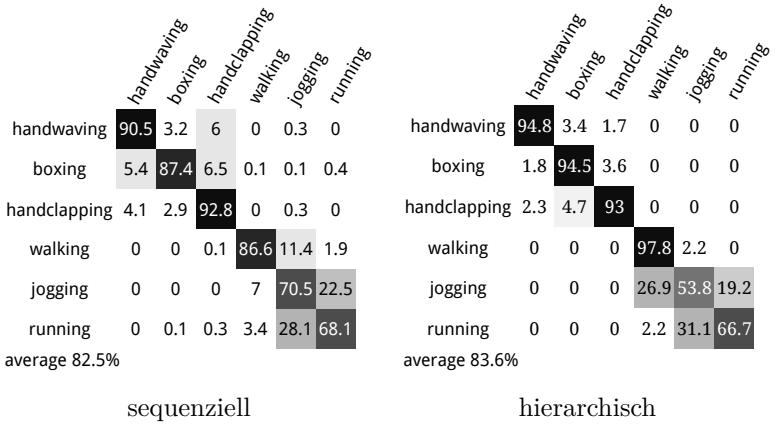


Tabelle 9.1.: Klassifikationsergebnisse der beiden Ansätze als *class confusion matrix*. Es wurde das Mittel aus 15 Durchläufen mit jeweils zufälliger Aufteilung des Test und Trainingsdatensatzes gebildet. Zur Ermittlung dieser Ergebnisse wurden die Parameter wie in Abschnitt 9.1 gesetzt.

nach dem *information gain* (Abschnitt 8.3), als auch die Erweiterung zum *semi-supervised* (Abschnitt 8.4) learning genutzt.

9.2. Ergebnisse

Die gemittelten Ergebnisse dieser 15 Läufe sind in Tabelle 9.1 aufgeführt. Beide Ansätze zeigen gute Klassifikationsleistungen und liegen insgesamt nur 1.1% auseinander. Der Sequenzielle Ansatz liegt dabei im Mittel 3.1% und der hierarchische Ansatz nur 2% hinter der *offline* Klassifikation nach Laptev. Tabelle 9.2 zeigt die Bandbreite der Ergebnisse der einzelnen Verfahren.

	Laptev	Sequentiell	Hierarchisch
avg	**85.6**	**82.5**	**83.6**
min	82.3	79.6	78.1
max	91.2	85.9	88.3
diff	8.9	6.3	10.2

Tabelle 9.2.: Bandbreite der Ergebnisse der einzelnen Verfahren. Alle Angaben in Prozent.

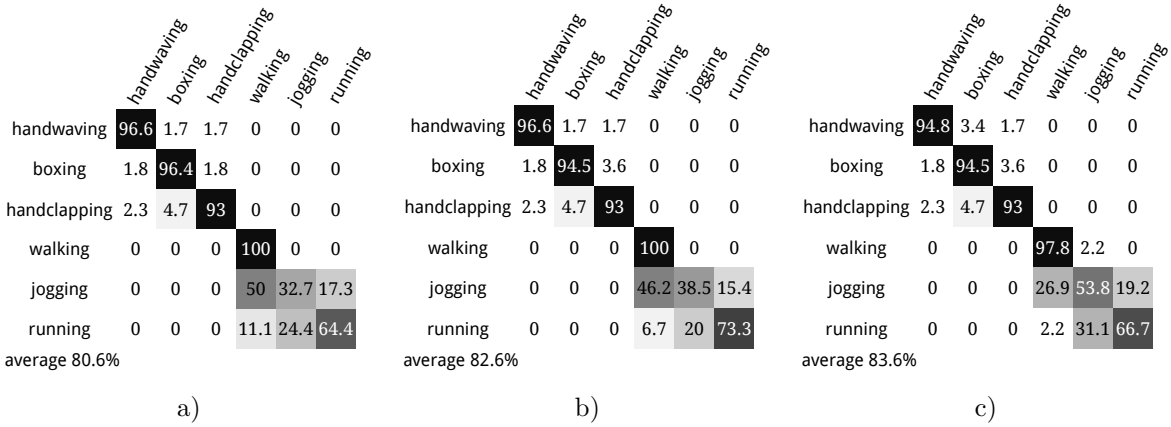


Tabelle 9.3.: Klassifikationsergebnisse des hierarchischen Ansatzes. a) ohne Gewichtung der Voten. b) mit Gewichtung der Voten. c) mit Gewichtung und *semi supervised learning*.

9.3. Diskussion der Ergebnisse

Die Ergebnisse der beiden entwickelten Ansätze zeigen, dass die *online* Klassifikation menschlicher Aktionen möglich ist, ohne merklich an Präzision der Klassifikation einzubüßen. Angesichts der besonderen Herausforderung, dass die beiden *online* Verfahren zu keinem Zeitpunkt eine globale Sicht auf die Daten haben, sondern immer nur einen Datenpunkt nach dem nächsten sehen, ist das Ergebnis durchaus gut. Der sequenzielle Ansatz zeigt dabei die gleichmäßigsten Klassifikationsergebnisse zwischen den einzelnen Aktionsklassen. Die Handaktionen werden scharf getrennt und die Fußaktionen zeigen in etwa die selbe Streuung wie bei dem *offline* Verfahren. Die Bandbreite der Klassifikationsergebnisse zwischen dem besten und dem schlechtesten Ergebnis ist mit 6,3% die geringste der drei getesteten Verfahren. Der hierarchische Ansatz zeigt das insgesamt bessere Klassifikationsergebnis. Die Handaktionen werden sehr scharf getrennt. Die Fußaktionen zeigen im Vergleich zum sequenziellen Verfahren eine deutlich höhere Streuung. Die Bandbreite der Ergebnisse ist bei diesem Verfahren mit 10,2% vergleichsweise hoch. Dieses Verfahren ist also im Vergleich zu den anderen etwas abhängiger von der Aufteilung der Trainings- und Testdaten.

Varianten des hierarchischen Ansatzes

In Tabelle 9.3 sind die Klassifikationsergebnisse aller Varianten des hierarchischen Ansatzes aufgeführt. Ohne die in Abschnitt 8.3 und 8.4 eingeführten Erweiterungen erreicht der Klassifikator eine *average precision* von 80,6%. Mit Gewichtung der Voten nach ihrem *information gain* werden 82,6% der Aktionssequenzen korrekt klassifiziert. Mit Nutzung des *information gain* und des *semi supervised learning* erreicht der Klassifikator 83,6%. Die

Gewichtung sorgt für deutlich bessere Trennung der Fußaktionen. Die visuellen Wörter, die bei vielen Aktionsklassen etwa in gleicher Häufigkeit vorkommen, gehen weniger Stark in die Klassifikation ein. Wie bereits in Abbildung 7.2 beleuchtet, haben die Fußaktionen nur wenige charakteristische Merkmale und profitieren damit besonders stark von der Gewichtung. Das *semi supervised learning* verbessert das Klassifikationsergebnis weiter. Die aus den besonders sicheren Prädiktionen ausgewählten Trainingsbeispiele helfen dem Klassifikator die Klassengrenzen der Fußaktionen genauer abzubilden. Da die maximale Anzahl an Neuronen für das Experiment festgelegt war und bei dieser Variante im Verhältnis mehr Neuronen in Bereichen des Eingaberaums waren, die mit Fußaktionen assoziiert sind, hat sich das Ergebnis der Handaktionen unwesentlich verschlechtert.

Kapitel 10

Fazit und Ausblick

Das Kernziel dieser Arbeit war es, ein *online* System zur Klassifikation menschlicher Aktionen zu entwickeln und zu evaluieren. Dazu wurden zwei verschiedene Ansätze vorgeschlagen und implementiert. Der erste Ansatz hat sich dabei an dem Aktionsklassifikationssystem von Laptev (Kapitel 3) orientiert. Dieser Klassifikator hat bereits gute Ergebnisse auf verschiedenen Aktionsdatenbanken gezeigt. Es hat sich ergeben, dass die Aktionsklassifikation auch unter den deutlich schwierigeren Rahmenbedingungen der inkrementellen *online* Klassifikation gut funktioniert. In dieser Arbeit wurden auf der Basis des von Laptev vorgeschlagenen Detektors und unter Nutzung des *bag* Modells zwei Verfahren zur *online* Klassifikation entwickelt und evaluiert. Es hat sich dabei gezeigt, dass beide Ansätze gute Ergebnisse liefern, die der *offline* Klassifikation nur wenig nachstehen.

Einsatz in der mobilen Robotik

Die in dieser Arbeit vorgestellten Ansätze zur Aktionsklassifikation eignen sich prinzipiell auch zum Einsatz auf mobilen Robotern. Der Harris3D Detektor arbeitet in der hier entwickelten Implementation auf den Aktionsvideos des KTH Datensatzes mit durchschnittlich 47fps auf einem PhenomII 955 Quadcore-Prozessor. Die Implementation skaliert gut auf mehreren Prozessorkernen, da die Berechnungen auf den einzelnen Skalenebenen unabhängig voneinander ausgeführt werden können. Auch eine Portierung auf aktuelle Grafikchipsätze ist denkbar und sollte sich mit vertretbarem Aufwand realisieren lassen. Mit der zunehmenden Verbreitung von Smartphones und Tablets kommen zunehmend auch leistungsfähige und dabei energiesparende Kombiprozessoren mit CPU- und GPU Kernen auf den Markt. Solche Systeme eigenen sich optimal für energiebeschränkte Einsatzgebiete wie die mobile Robotik.

Auch der GNG und der DYNG Algorithmus, auf denen die Ansätze *online* Klassifikation aufbauen, ließen sich grundsätzlich ebenfalls auf energiesparende Kombiprozessoren portieren. Bei diesen Algorithmen erfordert die Suche der nächsten Nachbarneuronen die meiste

Rechenzeit. Durch geschickte Partitionierung und Indexierung des Suchraumes ließe sich eine bessere Skalierung auf mehrere Recheneinheiten erreichen.

Durch die Nutzung rekursiver Filter zum Aufbau des Skalenraums wie in Abschnitt 4.3 auf Seite 36 vorgeschlagen, lässt sich die systembedingte Verzögerung bei der Detektion der *interest points* auf wenige *frames* reduzieren.

Weitere Aktionsdatenbanken

Der KTH Datensatz gilt als vergleichsweise einfacher Aktionsdatensatz, da die gezeigten Aktionen eine geringe Intra-Klassenvarianz aufweisen und die Aufnahmen vor homogenen Hintergründen gemacht worden sind. Das Klassifikationsverfahren nach Laptev wurde in [LMSR08] bereits auf einem komplexeren Datensatz mit Ausschnitten aus Hollywoodfilmen evaluiert. Dieser Datensatz enthält Videos aus 32 Filmen, die jeweils mit den 8 Aktionsklassen „AnswerPhone“, „GetOutCar“, „HandShake“, „HugPerson“, „Kiss“, „SitDown“, „SitUp“, „StandUp“ annotiert sind. Erste Versuche der in dieser Arbeit entwickelten Ansätze mit dem Hollywood Datensatz zeigten viel versprechende Ergebnisse und lassen erwarten, dass größere Datensätze der inkrementellen Vorgehensweise eher zuträglich sind. Da der Hollywood- und der nochmals erweiterte Hollywood2-Datensatz ganze Szenen mit teilweise mehreren darin vorkommenden Aktionsklassen enthält, müsste eine zeitliche Segmentierung der Szenen in einzelne Aktionssequenzen implementiert werden. Ein *sliding window* Ansatz, bei dem ein zeitliches Fenster fester Länge über die Videosequenz gelegt wird, könnte die gewünschte Segmentierung auf einfache Art und Weise realisieren.

Anhang A

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Die Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Bielefeld, den 26.04.2013

Panzner, Maximilian

Anhang B

Literaturverzeichnis

[BC11] Beyer, Oliver ; Cimiano, Philipp: Online labelling strategies for growing neural gas, 2011 (Proceedings of the 12th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2011)), S. 76–83

[BC12] Beyer, Oliver ; Cimiano, Philipp: DYNG: Dynamic Online Growing Neural Gas for Stream Data Classification, 2012

[BDSS01] Bobick, Aaron F. ; Davis, James W. ; Society, Ieee C. ; Society, Ieee C.: The Recognition of Human Movement Using Temporal Templates. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23 (2001), S. 257–267

[Fri95] Fritzke, Bernd: A Growing Neural Gas Network Learns Topologies. In: *Advances in Neural Information Processing Systems 7*, MIT Press, 1995, S. 625–632

[Gib50] Gibson, J.J.: *The perception of the visual world*. Houghton Mifflin, 1950

[HS88] Harris, Chris ; Stephens, Mike: A combined corner and edge detector. In: *In Proc. of Fourth Alvey Vision Conference*, 1988, S. 147–151

[KTH] *KTH Human Action Database. http://www.nada.kth.se/cvap/actions/*,

[Lap] Laptev, Ivan: CVPR08 - learning realistic human actions from video. `http://www.di.ens.fr/~laptev/actions/cvpr08.ppt`

[Lap04] Laptev, Ivan: Local Spatio-Temporal Image Features for Motion Interpretation. In: *KTH Numerical Analysis and Computer Science* (2004)

[Lap05] Laptev, Ivan: On Space-Time Interest Points. In: *International Journal of Computer Vision* 64 (2005), September, Nr. 2-3, 107–123. http://dx.doi.org/10.1007/s11263-005-1838-7. – DOI 10.1007/s11263–005–1838–7. – ISSN 0920–5691

[Lap13] Laptev, Ivan: *Space Time Interest Point Detector, Binary Implementation.* http://www.di.ens.fr/~laptev/download/stip-2.0-linux.zip. Version: 2013

[LCSL07] Laptev, Ivan ; Caputo, Barbara ; Schüldt, Christian ; Lindeberg, Tony: Local velocity-adapted motion events for spatio-temporal recognition. In: *Computer Vision and Image Understanding* 108 (2007), Dezember, Nr. 3, 207–229. http://dx.doi.org/10.1016/j.cviu.2006.11.023. – DOI 10.1016/j.cviu.2006.11.023. – ISSN 10773142

[Lin10] Lindeberg, Tony: *Generalized Gaussian Scale-Space Axiomatics Comprising Linear Scale-Space, Affine Scale-Space and Spatio-Temporal Scale-Space.* 2010. – 36–81 S. http://dx.doi.org/10.1007/s10851-010-0242-2. http://dx.doi.org/10.1007/s10851-010-0242-2. – ISBN 1085101002

[LK81] Lucas, Bruce D. ; Kanade, Takeo: An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI). In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, 1981, S. 674–679

[LMSR08] Laptev, Ivan ; Marszalek, Marcin ; Schmid, Cordelia ; Rozenfeld, Benjamin: Learning realistic human actions from movies. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition* (2008), Juni, 1–8. http://dx.doi.org/10.1109/CVPR.2008.4587756. – DOI 10.1109/CVPR.2008.4587756. ISBN 978–1–4244–2242–5

[MHK06] Moeslund, Thomas B. ; Hilton, Adrian ; Krüger, Volker: A survey of advances in vision-based human motion capture and analysis. In: *Computer Vision and Image Understanding* 104 (2006), November, Nr. 2-3, 90–126. http://dx.doi.org/10.1016/j.cviu.2006.08.002. – DOI 10.1016/j.cviu.2006.08.002. – ISSN 10773142

[Mor80] Moravec, H P.: *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*, Stanford University, Diss., 1980. http://www.mendeley.com/research/obstacle-avoidance-navigation-real-world-seeing-robot-rover/

[MS91] Martinetz, T. ; Schulten, K.: A "Neural-Gas" Network Learns Topologies. In: *Artificial Neural Networks* I (1991), S. 397–402

[Pop10] Poppe, Ronald: A survey on vision-based human action recognition. In: *Image and Vision Computing* 28 (2010), June, Nr. 6, 976–990. `http://doc.utwente.nl/70470/`

[Reu] Reuters, Alexei O.: *YouTube hits 4 billion daily video views. http://www.reuters.com/article/2012/01/23/usgoogleyoutube-idUSTRE80M0TS20120123*,

[WUK+09] Wang, Heng ; Ullah, Muhammad M. ; Kläser, Alexander ; Laptev, Ivan ; Schmid, Cordelia: Evaluation of local spatio-temporal features for action recognition. In: *British Machine Vision Conference*, 2009, 127