FAKULTÄT FÜR !INFORMATIK

Visual Information Retrieval: automatisierte Klassifikation von Snowboardclips

MAGISTERARBEIT

zur Erlangung des akademischen Grades
Magister

im Rahmen des Studiums
Wirtschaftsinformatik

eingereicht von
Dominik Lepizh
Matrikelnummer 0200908

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
DI Dr. Christian Breiteneder, Univ. Prof. und
DI Mag. Dalibor Mitrovic, Univ. Ass.

Wien, 27.05.2010

(Unterschrift Verfasser) (Unterschrift Betreuer)

Eidesstaatliche Erklärung

Dominik Lepizh
Dr. Neumanngasse 4/2
A-1230 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 27. Mai 2010 _____

Danksagung

An dieser Stelle möchte ich meinen Dank all jenen aussprechen, ohne die der Abschluss meines Studiums kaum möglich gewesen wäre.

Meinen Eltern und deren Lebenspartnern danke ich für ihr Vertrauen, das sie mir immer schenken, den Rückhalt, auf den ich mich immer verlassen kann und die Motivation, die stets zur rechten Zeit kam.

Besonderer Dank ergeht auch an *meine* Lebenspartnerin, die mich mit viel Geduld und Verständnis auf meinem Weg zum Magister begleitete, ohne die Ausbildung je in Frage zu stellen. Ebenso bin ich für ihre Lebensweisheit dankbar, von der ich täglich versuche zu lernen.

Diese wichtigsten Menschen in meinem Leben waren es, die mich überhaupt zu einem Studium ermutigten, meine Interessen förderten und stets bemüht waren finanzielle Hürden für mich so klein wie möglich zu halten.

Weiters danke ich natürlich auch all jenen, die mein Studium für mich so interessant und abwechslungsreich machten. Also Professoren, Assistenten und Studienkollegen - deren Zusammenkunft bei Projekten und Vorlesungen stets lehrreich, interessant und produktiv war.

Danke!

Abstract

Practical snowboard instructor training is an iterative process divided into two steps. In the first step, the future snowboard instructors are recorded on video performing on the slope. In the second step, these video recordings are analyzed and discussed with the focus on possible improvements of the future instructor's personal snowboarding style. The future instructors then try to apply the improvements in the next iteration of the first step.
This thesis presents a way of adequately supporting the second step by content-based classification and retrieval of snowboard videoclips.

The theory of snowboarding defines several turn types with different difficulties that are practiced step-by-step. Because rhythm and speed are the two main characteristics of different turn types, this thesis explores the feasibility to measure them via motion-detection and investigates how to deal with disturbing factors like camera shaking. The proposed method uses the output of optical flow analysis to compute the duration between two turns and the speed of the turn to classify the types of turns.
The audience in theoretical snowboard lessons is ususally bigger than one person, but everyone needs individual feedback during analysis. As a result it is very important for trainers to be able to quickly present appropriate video samples - either from the same or from another person. This personalized feedback motivates the second presented method in this thesis. This method employs an established color analysis technique to distinguish which person is shown in the videoclips. The method enables trainers to select individual videoclips for presentation.

In order to evaluate the acquired techniques and developed methods, they are applied on a manually generated test-set of videoclips which were recorded during several days of training by this thesis' author.
Turn type classification yields good results in computing the average number of frames between two shifts in direction (wide driven carving turns versus fast moving short turns) so 85% percent of videoclips are classified correctly (65% even clearly).
The distinction of videoclips based on depicted persons is highly dependent on scenery and illumination, which disturbs classification results because color-matching fails (classification error-rate rises linearly with the number of analyzed videoclips).

Zusammenfassung

Bei der Analyse von Videoaufnahmen im Rahmen eines Snowboard-Lehrbetriebs stellt sich immer wieder das Problem, dass zur Diskussion passende Stellen am Videoband gesucht werden müssen. Einmal sucht man ähnliche Abfahrtsszenen (im Bezug auf Schwungart und Fehler) einer zweiten Person, kurz darauf sucht man weitere Abfahrten der gleichen Person.
Die vorliegende Arbeit entwickelt eine entsprechende inhaltsbasierte Klassifikation von Snowboard-Videoclips und konzentriert sich dabei auf die beiden Merkmale Schwungart und Personenerkennung.

Im Lehrplan des Snowboardunterrichts sind Grundschwungarten definiert, die unterschiedliche Schwierigkeitsgrade beinhalten und aufbauend geschult werden. Da dabei Rhythmus und Geschwindigkeit die zwei Hauptunterscheidungsmerkmale sind, untersucht diese Arbeit die Möglichkeit, die beiden Merkmale mittels Methoden der Bewegungserkennung zu messen und mit Störfaktoren wie Verwackelungen umzugehen. Dabei bedient sich die vorgestellte Methode der Optical Flow Analyse, um den Rhythmus - also die mittlere Dauer zwischen zwei Schwüngen - zu berechnen und so eine Klassifizierung zu ermöglichen. Die Zahl der Kursteilnehmer ist meistens recht hoch, trotzdem benötigt jeder Teilnehmer ein individuelles Feedback zu den aufgezeichneten Abfahrten. Daher ist es für den Trainer sehr wichtig schnell andere Anschauungsbeispiele parat zu haben, die seine Argumente bildlich unterstützen. Die zweite vorgestellte Methode stellt - anhand einer farbanalysierenden Technik - die Möglichkeit zur Verfügung, Videoclips nach Personen zu klassifizieren und so eine individuelle Clipselektion zu ermöglichen.

Um die verwendeten Techniken und entwickelten Methoden zu evaluieren, wurden sie an einem Test-Set von Videoclips angewandt und deren Ergebnisse beurteilt. Das notwendige Filmmaterial wurde dabei an Trainingstagen im Schnee aufgenommen und später für die Aufgabe dieser Arbeit von Hand selektiert. Die Klassifikation nach Schwungarten liefert bei der Berechnung der mittleren Bildanzahl (= Dauer) zwischen zwei Richtungswechseln sehr gute Ergebnisse, wodurch 85% der Testclips korrekt klassifiziert werden (65% sogar sehr klar). Der Bildvergleich aufgrund farblicher Aspekte (Personen) ist allerdings sehr von der Szenerie und den Lichtverhältnissen abhängig was zu erheblichen Fehlern in der Klassifizierung führt (Fehlerrate steigt linear mit der Anzahl an analysierten Clips).

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Fotos, Musik und Videos - mit der Digitalisierung und zentralen Verwaltung all unserer multimedialen Inhalte sammelt sich ein stetig wachsender Datenberg an, dessen einzelnen Inhalte später in irgendeiner Form schnell wiedergefunden werden sollen. Die Voraussetzungen bzw. notwendigen Maßnahmen dafür sind aber meist zeit- bzw. ressourcenraubend.
Zu Beginn der digitalen Revolution im Foto- und Videobereich war es Sache der bloßen, individuellen, textuellen Beschreibung der Medien, die eine Katalogisierung und strukturierte Datenhaltung ermöglicht. Dabei werden sukzessiv neue Medien der Sammlung hinzugefügt und entsprechende Beschreibungen (Einteilung in Kategorien, Suchbegriffe, Datum, GPS, uvm.) meist händisch beigefügt. Diese so genannten Metadaten, also beschreibende Daten zu vorliegenden (Medien-) Inhalten, sind aber nur dann ein gutes Verwaltungsinstrument, wenn der (Zeit-) Aufwand bei der Erfassung keine Rolle spielt.

Denkt man aber an die Anforderungen der Medien-Verwaltung bei Unternehmen (Fernsehen, Rundfunk, Presse,...), deren Datenaufkommen das private um ein Vielfaches übersteigt und Zeit ein wesentlicher Erfolgsfaktor sein kann, so stößt man bei manueller Medien-Verwaltung schnell an Grenzen.
Es deutet also vieles auf die Notwendigkeit einer automatisierten Kategorisierung von multimedialen Inhalten nach unterschiedlichen Gesichtspunkten hin. So ist es im Zeitalter von digitalen Texten keine Besonderheit mehr, dass nicht nur Presseunternehmen ihre Publikationen automatisiert verwalten und wiederauffindbar machen. Die Auswertung eines textuellen Inhalts (bzw. die Suche nach Teilen davon) ist nicht zuletzt durch die Internettechnologie ohne Probleme möglich und prägte den Begriff der „Informationsgesellschaft“.

Die strukturierte Archivierung von Fotos, Musik und Videos hingegen stellt schon wesentlich höhere Anforderungen an die Systeme. Die zugrunde lie-

genden Daten sind zwar alle interpretier- und verwertbar, die Techniken und Methoden dafür aber bei Weitem noch nicht so zuverlässig und mächtig, wie wir das bei textuellen Inhalten schon gewohnt sind.
Erste konkrete Anwendungen in diesem Bereich sind Suchalgorithmen/-systeme, die aufgrund eines Ausgangsmediums (Foto/Video/Musik) ähnliche Inhalte aus der Datenbasis zurückliefern (zum Beispiel Fingerprint-Suche in der Verbrecherdatenbank). Solche Systeme beruhen auf der automatisierten Extraktion und Auswertung von beschreibenden Daten. Bei visuellen Medien sind dies vor allem farbliche, geometrische, strukturelle und zeitliche Eigenschaften. So ist es mit speziellen Systemen bereits möglich in einer Mediendatenbank nach gewissen Fotomotiven (z.B. einem Sonnenuntergang) zu suchen. Genauso können die einzelnen Bilder eines Videoclips (= frames) analysiert und nach gewissen Gesichtspunkten durchsucht werden.
Natürlich können aber nicht alle Metadaten automatisiert erzeugt werden. Die manuelle Erfassung von gewissen Zusatzinformationen zu einem Video oder einem Foto wird immer erforderlich bleiben. Denn dass ein System in naher Zukunft in der Lage ist (mit vertretbarem Aufwand) selbsttätig den Namen der interviewten Person oder sogar die wichtigsten Schlagworte des Interviews herauszufiltern (beispielhaft), ist eher unwahrscheinlich. Daher wird man so bald auf die manuelle „Beschriftung“ von Medien durch einen Menschen nicht verzichten können.
Neben der gestalterischen (digitalen) Bearbeitung von multimedialen Inhalten ist also eine intelligente Archivierung derselben unerlässlich. Nur wenn gewährleistet ist, dass wir das Gesuchte im Vorhandenen schnell finden, können wir sinnvoll mit den Gigabyte an Fotos/Videos arbeiten - egal ob im privaten oder kommerziellen Bereich. Diesem Problem ist auch diese Arbeit gewidmet, die versucht eine sehr spezielle Ausprägung davon zu lösen.

1.1 „Visual Information Retrieval“ (VIR)

Bereits Anfang der 1980er Jahre hat man dieses Problem erkannt und angefangen Lösungen dafür zu entwicklen. Die daraus entstandene Forschungsdisziplin nennt sich „Visual Information Retrieval“ (kurz VIR). Die Kernaufgabe des VIR ist die Gewinnung relevanter Informationen aus visuellen Daten und findet in immer mehr und unterschiedlicheren Gebieten der elektronischen Datenverarbeitung Anwendung. Sei es im Verkehrsüberwachungs- (SectionControl, Stausensoren,...) oder Gesundheitswesen (Computertomographie,...); computerunterstützte Analyse von visuellen Daten eröffnen völlig neue Arten der Information und können traditionelle Analyse- und Suchverfahren optimieren.

1.1.1 Aufgaben und Ziele

Der Umstand, dass der bloße, pixelweise Vergleich von visuellen Medien sehr aufwändig und zudem kaum aussagekräftig ist, macht es erforderlich, im Vorfeld Gesichtspunkte bzw. Eigenschaften zu definieren, die von Interesse sind und später auf alle vorliegenden Medien angewandt werden können.
So ist es für den Endanwender wenig hilfreich Suchanfragen an das Medien-Managementsystem anpassen zu müssen - z.B. „finde alle Fotos, deren ersten 1000 Pixel einen Rotanteil von 70-90% haben“ (so könnte etwa die Suche nach Sonnenuntergangsszenen aussehen) - denn erstens entspricht dies nicht dem menschlichen Zugang zu Foto-/Videoinhalten und zweitens würde eine entsprechende pixelweise Auswertung aller vorliegenden Medien (zur Laufzeit) viel zu lange dauern.

Zur Gewinnung inhaltsbasierter Daten werden daher lt. VIR so genannte „Features“ definiert. Features - also Eigenschaften - eines Mediums können sehr vielschichtig sein; klassische Bildfeatures sind jedoch Farbhistogramme (Informationen zur Farbverteilung im Bild), Struktur- und Textureigenschaften. Jedem Feature liegt ein Extraktionsalgorithmus zu Grunde, der das

Bild-/Videomaterial analysiert und die gewünschten Informationen (meist verdichtet) zur Verfügung stellt. Die Ergebnisse sind meist Zahlenvektoren, die als Punkte im Vektorraum gesammelt werden können [33] und dadurch Distanzberechnungen (zu Ergebnissen anderer Medien) erlauben.

Das VIR unterscheidet prinzipiell zwei Arten von Features [5]:

quantitative (low-level) Features umfassen alle Eigenschaften, die sich leicht in Zahlen ausdrücken lassen. Die zuvor genannten Features (Farbhistogramme, Struktur/Textur) sind quantitativer Natur; ihre Ausprägungen werden zahlenmäßig ermittelt und können dementsprechend einfach weiterverarbeitet oder verglichen werden. Diese low-level-Features sind, sofern der Algorithmus einmal entwickelt wurde, sehr einfach zu extrahieren und bieten eine erste Grundlage zur inhaltsbasierten Beschreibung von Medien.

qualitative (high-level) Features hingegen sind Eigenschaften, für die semantisches Zusatzwissen benötigt wird. So sind den Wahrnehmungsfähigkeiten heutiger Computersysteme weiterhin Grenzen gesetzt, die es unmöglich machen das menschl. Wissen bzw. Wahrnehmung zu ersetzen. Die einzige Möglichkeit so genannte high-level-Features zu schaffen ist das manuelle Anreichern von low-level-Features durch semantische Informationen. Dabei sind vor allem drei Informationsquellen ausschlaggebend:

Informationen zur Anwendungsdomäne: etwa das Einschränken der Suche nach Objekten in einem Bild auf vordefinierte Muster/Formen/Farben, von denen man weiß, dass sie relevant sind. Dies spart Rechenzeit und liefert auf eine bestimmte Fragestellung zugeschnittene Ergebnisse.

Informationen zum Benutzer: klassische Zielgruppen-Informationen wie Alter, Geschlecht, Herkunft sind ebenfalls Faktoren, die eine Auswertung informativer und verständlicher für den Betrachter gestalten.

Informationen zu den semantischen Eigenschaften eines Features: Kein Computersystem kann, ohne dass es ihm zuvor beigebracht wurde, menschliche Empfindungen (wie unterbewusste Reaktionen auf Farben) wahrnehmen oder nachahmen. So ist eine Suche nach romantischen, warmen Fotoszenen nur dann möglich, wenn man dem System im Vorfeld bekanntgegeben hat, dass vor allem Rottöne beim Menschen warme Gefühle auslösen.

Um Ergebnisse von inhaltsbasierten Suchanfragen für Menschen möglichst aussagekräftig zu gestalten, bedarf es daher der Anreicherung von Medien (-systemen) um die Bedeutung gewisser Themen/Tatsachen für den Menschen; die oft auch subjektiv und unterschiedlich ausfallen kann.

1.1.2 Prozess der Informationsgewinnung

Ausgehend von den definierten Features müssen natürlich Algorithmen/Verfahren entwickelt werden, die in der Lage sind, diese Informationen aus den Medien zu extrahieren. Jedes Medium, das einer Mediendatenbank dann hinzugefügt wird, wird den Verfahren entsprechend analysiert und die Ergebnisse als zusätzliche Metadaten verspeichert. Daraus ergibt sich eine „Wolke“ an Daten, die - wie zuvor schon erwähnt - als Punkte im Vektorraum interpretiert und verglichen werden können.
Dem Benutzer können dann möglichst ähnliche Medien seines Suchobjekts angeboten werden, der in weiterer Folge die Möglichkeit haben sollte, für ihn relevante bzw. nicht-relevante Medien markieren zu können und die Suche damit zu verfeinern. Dieses, vom VIR vorgeschlagenen, Grundkonzept nennt man *relevance feedback* (siehe auch Kapitel 2.3) - also ein Feedback des Users an das System zur Verfeinerung des Suchergebnisses. Dadurch entsteht ein iterativer „Refinement“-Prozess zwischen User und System, der als Ziel ein möglichst treffendes Suchergebnis hat.

Beim Konzept des *„kernel-based learning“* wird das Verhalten bzw. die Eingaben des Benutzers beim Refinement vom System ausgewertet und gespeichert, um bei späteren Suchanfragen diese Erkenntnisse miteinfließen zu lassen.

1.1.3 Anwendungsgebiete

Neben den bereits erwähnten stellt eines der jüngeren Anwendungsgebiete der Sport und dessen Umfeld dar. Trotz der relativen Jugendlichkeit ist aber auch hier schon der Einsatz von VIR-Konzepten weit fortgeschritten und so werden beispielsweise bei Ballsportarten diffizile Entscheidungen aufgrund computergestützer Analysen von Spielzügen getroffen oder Auswertungen über den Laufstil eines Langstreckenläufers angestellt [1].

1.2 Motivation

Neben meinem Studium der Wirtschaftsinformatik an der TU Wien habe ich im Winter 2005/06 die Ausbildung zum Snowboard-Instruktor an der Bundesanstalt für Leibeserziehung (BAFL Graz) begonnen und später erfolgreich absolviert. Während des Schneekurses wurden wir Anwärter immer wieder gefilmt, um am Abend bei der Videoanalyse gemeinsam Fehler/Schwächen zu identifizieren und spezielle Übungen/Tipps (für den nächsten Tag) vom Kurstrainer zu erhalten. Dabei sammelte sich ein recht umfangreiches Videomaterial an, welches mir nach dem Kurs weiter zur Verfügung stand.

Während der täglichen Videoanalyse ging jedoch sehr viel Zeit damit verloren, dass immer wieder Stellen im Video (auf Band in DigiCam) gesucht werden mussten. Einmal wollte man ähnliche Szenen der gleichen Person zur weiteren Diskussion parat haben, ein anderes Mal suchte man nach positiven Bewegungsabläufen bei anderen Personen (um Unterschiede genau aufzeigen zu können).

Das „Spul- bzw. Suchproblem“, mein Interesse am Themengebiet des VIR und das vorhandene Videomaterial, ließen die Vision eines Systems (Software-Tools) entstehen, das beim Sichten von derartigen Videos hilft und gewisse Funktionalitäten zur Verfügung.

Der technische Fortschritt in den letzten Jahren im Bereich der consumer electronics kommt diesem Vorhaben deutlich entgegen. So ist die DigiCam-Technologie heute schon so ausgereift, hochqualitativ und trotzdem winzig klein, sodass solche Anwendungen auch im privaten (Freizeit-) Bereich letztendlich gefordert und genutzt werden. Digitale Videos in HD-Qualität werden heute gleich direkt auf Festplatten aufgenommen, wodurch einerseits langwieriges digitalisieren und andererseits der Qualitätsverlust dabei entfällt. Gerade die höhere Qualität kommt den sensiblen Feature-Berechnungen entgegen, da mit weniger Bildrauschen und störenden Bildfehlern zu rechnen ist.

Die Motivation dieser Arbeit wird daher vom Problem des vorliegenden, nicht kategorisiertem/katalogisiertem Videomaterials getrieben und hat einen effizienteren Schulungsbetrieb bzw. im Speziellen eine Videoanalyse mit moderner Computerunterstützung als Ziel.
Aus pädagogischer Sicht ist natürlich der viel flüssigere und im Endeffekt auch reichhaltigere (im Sinne von mehr Anschauungsbeispielen in der gleichen Zeit) Unterricht die Motivation für ein solches Vorhaben.

Die Hauptkomponenten eines entsprechenden Systems sind einerseits ein sehr minimales, benutzerfreundliches Userinterface und andererseits die vollautomatische Klassifizierung von Videoclips.
Die vorrangigen Vorteile im Schulungsbetrieb wären:

1. der Wegfall lästiger Spul-/Suchzeiten, da direkter Zugriff auf gewünschte Clips ermöglicht wird, und

2. zum anderen könnte man so recht schnell Vergleiche zwischen Bewe-

gungsabläufen dar-/anstellen (die Möglichkeit gleichzeitiger Präsentation mehrerer Clips (durch das System) vorausgesetzt) und so Unterschiede im Fahrstil deutlich aufzeigen.

Weiters ist der Umstand sehr interessant, dass die Clips alle in eine Datenbank zusammenlaufen und diese Tag für Tag wächst. Das Resultat ist eine wertvolle Sammlung von katalogisierten Übungsfahrten, die eine Analyse der individuellen Historie des Lernfortschritts möglich macht.

1.3 Abgrenzung

Neben der Digitalisierung (analogen Filmmaterials), Archivierung und Präsentation des Videomaterials (was alles zwar zum Endprodukt zweifelsfrei dazu gehören müsste, aber den Rahmen dieser Magisterarbeit sprengen würde), ist die technische Klassifizierung einzelner Szenen nach gewissen Gesichtspunkten (siehe 1.4), der Teil dem diese Magisterarbeit gewidmet sein soll.

1.4 Problemdefinition

Wie schon zuvor angedeutet, beschränkt sich der praktische Teil dieser Magisterarbeit auf jenen Teil der „Vision“, der für die Analyse und Kategorisierung der Clips verantwortlich ist.
Zu planen/entwickeln ist also eine Logik, die einen vorliegenden Videoclip mit Merkmalen „beschriftet“ um später eine Selektion zu ermöglichen. Dafür sollen bestehende Methoden/Verfahren im Bereich der VIR recherchiert, kombiniert und gegebenenfalls erweitert werden.

Neben anderen sind dabei folgende Merkmale für eine effiziente Videoanalyse (Unterricht) relevant, für die in dieser Arbeit eine entsprechende Lösung der Kategorisierung gefunden werden soll:

1.4.1 Kategorisierungsmerkmal: Person

Da man davon ausgehen kann, dass an einem Snowboardkurs mehrere Schüler teilnehmen, ist das Merkmal Person von hoher Relevanz. Denn sollen schnell alle Pistenabfahrten einer Person (zur Veranschaulichung) vorliegen, muss das System diese zuvor analysiert und zugeordnet haben. So könnten auch Einzelanalysen (also nicht in der Gruppe) schneller/bequemer durchgeführt werden, da nicht immer wieder über Szenen hinweg gespult werden müsste. Die virtuelle Aneinanderreihung aller Clips einer Person wäre hier eine Zielfunktionalität.

Da sich eine Person im Schnee hauptsächlich durch die Farben ihrer Kleidung von anderen unterscheidet (vgl. Beispiele in Abb. 1), sollte diese Aufgabe mit Hilfe von Farbvergleichen (täglich gleiche Kleidung vorausgesetzt) zu bewerkstelligen sein.

Abbildung 1: Merkmal: Personen - Farbunterschiede

1.4.2 Kategorisierungsmerkmal: Schwungart

Die zweite interessante Dimension in diesem Anwendungsgebiet sind Schwungarten.

fachlicher Exkurs [29][28]: Snowboard-Schwungarten sind im Prinzip aufbauende Schwierigkeitsgrade bzw. Arten einen Schwung im Schnee durchzuführen. Die wichtigsten Schwungarten sind dabei der Anfänger-, Kurz- und Carvingschwung. Diese werden lt. Schulungsplan aufbauend und abhängig vom Lernfortschritt der Reihe nach gelehrt. Beim Lehrbetrieb kommt es darauf an, dass der Instruktor/Lehrer auf eine lupenreine Vorführung achtet, da Schüler vor allem Gesehenes gut umsetzen können. Das Abschauen, Einprägen und Nachahmen bestimmter Bewegungsabläufe sind die zentralen Lernwerkzeuge der Schüler. Das Analysieren der eigenen Bewegung mit Hilfe einer Videoaufnahme dient dabei die eigenen Schwächen selbst zu erkennen und später gezielt zu trainieren.

Zur Kategorisierung von gefilmten Abfahrtsszenen sollte das System daher eigenständig nach Schwungarten unterscheiden können. Dann ist es auch

schnell möglich, Fahrstil-Vergleiche unterschiedlicher Personen anzustellen.

Abgrenzung

Einen Schwung automatisiert auf seine korrekte Ausführung hin zu analysieren ist vielleicht möglich, jedoch nicht Aufgabe dieser Arbeit. Die fachlichen Hintergründe wurden nur deswegen erwähnt, um die Notwendigkeit einer Klassifizierung nach Schwungarten zu rechtfertigen.

Für eine sinnvolle Umsetzung ist es weiters relevant, dass sich wahrscheinlich nicht alle fünf Schwungarten automatisiert eindeutig unterscheiden lassen. Neben einer Vielzahl an marginalen Unterscheidungsmerkmalen, sind für die Grobeinteilung (und dieses Software-Vorhaben) vor allem die Fahrgeschwindigkeit, der Rhythmus (in dem Schwünge aneinander gereiht werden) und eventuell die Oberkörperaktivität ausschlaggebend.

Mit diesen zwei/drei Merkmalen muss nun versucht werden, Mechanismen zu entwickeln um eine näherungsweise Klassifizierung in etwa zwei Kategorien durchführen zu können.

1.5 Einbettung in das Forschungsumfeld

Schon alleine die zwei gerade definierten Aufgaben zeigen die Reichweite der Visual Information Retrieval und dessen Umfeld.
Bei der einen geht es vor allem um das Extrahieren und Auswerten von Bewegungsdaten aus bewegten Bildern (Videos); bei der anderen um die Segmentierung von Bildregionen und den farblichen Vergleich der selben.

Es sollen bereits vorliegende Konzepte untersucht und zu einer befriedigenden Lösung der Probleme miteinander kombiniert und ggf. erweitert werden.

Vordergründig beteiligte Forschungsgebiete dabei sind (siehe auch Kapitel 2)

Motion Detection Möglichkeiten zur Extraktion von Bewegungsdaten aus bewegten Bildern (Optical Flow, Stereo matching,...),

Image Stabilization Stabilisierung von verwackelten Videoszenen,

Color Features zur Differenzierung von unterschiedlichen Personen und

Region/Foreground Segmentation Snowboarder vom Schnee trennen.

1.6 Ziel der Arbeit

Im Zuge dieser Magisterarbeit soll zum einen der state-of-the-art im VIR zu diesen Teilbereichen recherchiert, geeignete Konzepte/Methoden zusammengetragen und in einem Prototypen umgesetzt werden.
Bei diesem Prototyp kann es sich jedoch noch nicht um eine voll funktionstüchtige, einsatzbereite Software für SB-Instruktoren handeln; sondern es soll versucht werden, das Kernproblem - die Klassifizierung von Videoclips - so gut wie möglich zu lösen und für eine entsprechende Demonstration (rudimentäres User Interface) zu sorgen.
Natürlich sollen die Ergebnisse kritisch analysiert/evaluiert und entsprechend begründet werden.

2 Analyse bestehender Ansätze

Wie schon in 1.5 erwähnt, erfordert das Ziel dieser Arbeit eine intensive Auseinandersetzung mit relevanten Forschungsthemen und eine Kombination von bestehenden Methoden/Techniken.

2.1 Image Information Retrieval

Grundlage von Methoden, deren Ziel die inhalts-basierte Beschreibung von Medien ist, sind elektronisch messbare Attribute multimedialer Inhalte, die sich bei zweidimensionalen Bildern grundsätzlich in zwei Arten unterteilen lassen:

- räumlich-bezogene Attribute (spatial information) umfassen jene Bildinformation, die in Abhängigkeit zur Anordnung der einzelnen Bildpixel im Koordinatensystem stehen; sie umfassen Informationen zur Form und Textur von Objekten.
- räumlich unabhängige Attribute (non-spatial) hingegen beschreiben Bilder mit Hilfe statistischer Merkmale; zum Beispiel mittels Farbinformation (Histogramme).

2.1.1 Farbe

Farbe spielt beim Vergleich von Bildern eine wesentliche Rolle. Farbe im elektronischen Sinne definiert sich über den jeweiligen Anteil der drei Grundfarben (Rot, Grün, Blau) in einem Pixel. Dabei haben sich unterschiedliche Farbsysteme, also Varianten eine bestimmte Farbe zu beschreiben, etabliert. Neben dem wohl bekanntestem RGB-Farbsystem, wurden viele Alternativen dazu entwickelt (wie CMYK, HSV, YUV, etc.), die alle versuchen die Schwächen von RGB zu minimieren oder spezielle Anwendungsgebiete zu bedienen. Bei diesen definiert sich eine Farbe nicht - wie bei RGB - über

das Verhältnis der drei Grundfarben, sondern - anhand des Beispiels HSV - durch die Werte Farbton, Farbsättigung und Dunkelstufe.

Farbhistogramme können zur optischen Beschreibung eines Bildes/Videos herangezogen werden. Dabei gibt es Auskunft über die Farbverteilung innerhalb des Bildes - die Anzahl der Pixel einer bestimmten Farbe (eines def. Farbraums). Hat man Farbhistogramme von zwei versch. Bildern kann zumindest ein grober Vergleich der beiden aufgrund deren Farbverteilung vorgenommen werden [37].

Verfahren: Zu Beginn werden so genannte *„bins“* (= Container oder Boxen) definiert, die nichts anderes als Farbkategorien darstellen, nach denen der Pixel-Zählvorgang sein Ergebnis trennt. Ein *bin* kann ein einzelner Farbwert oder eine Gruppe/Intervall an Farbwerten sein. Eine Kostenfunktion entscheidet dann bei der Analyse für jedes Pixel, welchem *bin* es zugeordnet wird. Das Ergebnis dieser Featureextraktion kann dann mit den Werten anderer Images verglichen werden, sofern die gleiche *bin*-Struktur zur Anwendung kam oder diese mit entprechenden Techniken (wie in [31]) vereinheitlicht/normalisiert wurde.
Wenn I_R, I_G und I_B normalisierte Farbhistogramme eines Bildes in der Bilddatenbank und Q_R, Q_G und Q_B die eines Suchbildes sind, so ist die Ähnlichkeit beider - über Histogramm-Verschneidung (Intersection) - wie folgt definiert [12]:

$$S_c^{HI}(I,Q) = \tag{1}$$

$$\frac{\Sigma_r min(I_R(r),Q_R(r)) + \Sigma_g min(I_G(g),Q_G(g)) + \Sigma_b min(I_B(b),Q_B(b))}{min(|I|,|Q|) * 3}$$

Alternativ dazu kann die Distanz zwischen zwei vorliegenden Farbhistogram-

men auch mit Hilfe der euklidischen Distanz berechnet werden:

$$S_c^{ED}(I,Q) = \tag{2}$$

$$1 - \sqrt{\frac{\Sigma_r(I_R(r) - Q_R(r))^2 + \Sigma_g(I_G(g) - Q_G(g))^2 + \Sigma_b(I_B(b) - Q_B(b))^2}{2 * 3}}$$

Wie bei allen Distanzmaßen gilt, je kleiner die Differenz, desto größer die Ähnlichkeit der beiden verglichenen Bilder.
Der große Vorteil beim Vergleich von Farbhistogrammen ist, dass sie robust gegenüber Rotations- und Translationsbewegungen sind. Sie führen also auch zu einem positiven Matching wenn das Motiv im Bild kopfüber fotografiert wurde (Rotation).

Einschränkungen: Gleichzeitig gibt es in der Verwendung von Farbhistogrammen aber Einschränkungen. Denn wird versucht aufgrund eines Referenzhistogramms ähnliche Bilder in einer Datenbank zu finden, werden sehr wahrscheinlich auch Bilder dabei sein, die zwar ein ähnliches Farbhistogramm aufweisen, jedoch für den Menschen unterschiedliche Bedeutung haben [36]. Das ist die große Schwäche dieser Ähnlichkeitsanalyse und sollte daher - wenn alleine angewandt - durch ein kluges „Refinement"-Verfahren ergänzt werden, um dem System beizubringen, welche Suchergebnisse für den User relevant sind und welche nicht (siehe auch Kapitel 2.3.1).
An Mächtigkeit gewinnt diese Art von Bildvergleich erst in Kombination mit Textur- und/oder Form-Vergleichen (z.B. der menschl. Gestalt, etc.), da hier der Suchalgorithmus intelligenter gestaltet wird und auf zusätzliche Merkmale Rücksicht nimmt.

Speziell im vorliegenden Anwendungsgebiet (Snowboardvideos) würde die Suche nach einer bestimmten Person, wenn sie nur auf einem Farbhistogramm basiert, wahrscheinlich alle Videoclips als Ergebnis zurückliefern. Ein

Bild bzw. dessen Farbverteilung unterscheidet sich bei einem Snowboarder auf weißem Schnee nämlich zu wenig, um hier nur mit einem Farbhistogramm befriedigende Resultate zu erhalten.
Daher sollte zuvor die Form bzw. die Region angegeben/erkannt werden, die verglichen werden soll; nämlich nur die Person (Kleidung) - siehe 2.2.3.

Dominant Color Descriptor (MPEG-7) 2002 veröffentlichte die MPE-Group den - später als ISO-Standard verabschiedeten - Standard MPEG-7 [19] [35]. Dieser umfasst eine Reihe an Definitionen in den Bereichen Farbe, Textur, Form, Bewegung, Audio, etc., die der Beschreibung von multimedialen Inhalten dienen.
Neben den Descriptoren (Feature-Beschreibungen) und -schemen beinhaltet der Standard eine Sprache (Descripton Definition Language) mit der die Descriptoren erweitert und angewandt werden können.

Ein an dieser Stelle sehr interessanter Descriptor ist der Dominant Color Descriptor (DCD). Dabei geht es um die Extraktion des/r Dominant Color(s), also der dominierenden Farbe(n) im Bild, deren Information genauso als Input für einen Farbvergleich dienen kann.
Kernkomponente dieses Features ist die Anwendung eines Cluster-Verfahrens auf die Bilddaten, das (in diesem Fall) Farbansammlungen (= Cluster = Datenhaufen) identifiziert und so Aufschluss über dominierende Farben gibt.

Verfahren: Zu Beginn wird dem Verfahren mitgeteilt, wieviele Dominant Colors gefunden werden sollen, oder anders formuliert: in wieviele Container (= Dominant Color) alle Pixel eines Images getrennt werden sollen. Sollen beispielsweise die *zwei* dominierenden Farben ermittelt werden, sucht sich das Verfahren ausgehend von zwei zufällig gewählten Startpunkten (im Farbraum) (vgl. schwarze Punkte in Abbildung 2) mit jeder Iteration einer Zentroid-Berechnung, bei der die Datenpunkte aufgrund eines Distanzma-

ßes und einer Kostenfunktion einem der beiden Clusterzentren zugeordnet werden, das neue Zentrum je Cluster und rückt den Startpunkt für den nächsten Iterationsschritt dorthin (Verlauf siehe schwarze Linien in Abbildung 2). Nach einem finalen Iterationsschritt (beliebige Abbruchbedingung) stehen die endgültigen Zentren der `n` Cluster fest (weiße Punkte) und bestimmen nun die (statistisch) mittlere Ausprägung der dominierenden Farben (hier zwei: grüne und graue Datenpunkte - oberhalb und unterhalb der roten Linie).

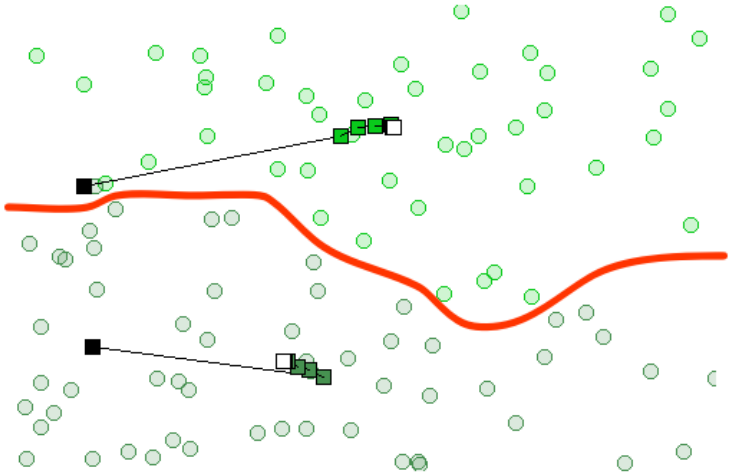

Abbildung 2: Cluster-Verfahren: Visualisierung der Iterationsschritte anhand eines Beispiels

Einer der Vorteile dieses Verfahrens ist, dass das Ergebnis nicht durch zuvor definierte Werte im Farbraum beeinflusst wird oder an einen bestimmten Farbraum gebunden ist. Als Vergleich dazu: bei traditionellen Farbhistogrammen muss zuvor definiert werden in welche Farbklassen unterschieden werden soll; beim DCD werden diese Information zur Laufzeit ermittelt und danach ausgewertet. Der berechnete Descriptor wird dabei in folgender Form bereit-

gestellt:

$$DCD = \{(c_i, p_i, v_i), s\}, i = 1, 2, ..., N \tag{3}$$

wobei

$\boldsymbol{N}$ Anzahl der dominierenden Farben; jedoch auf 8 limitiert

c_i Farbvektoren; beschreiben die dominante Farbe

p_i Prozentzahl; gibt Auskunft über die Anzahl der Pixel in dieser Farbe

v_i Varianz der Farbe im Image

s räumliche Koherenz aller dominanter Farben

Ähnlichkeitsvergleich Das Ergebnis des DCD kann weiters als Input für einen Ähnlichkeitsvergleich zwischen Images verwendet werden. MPEG-7 definiert für dieses Matching die Dominant Color Distance, bei der die Deskriptoren zweier Images auf ihre Distanz hin verglichen werden, und deren Formel vereinfacht folgendermaßen beschrieben werden kann:

$$Difference(D_1, D_2) = W_1\, SC_{diff}\, DC_{diff} + W_2\, DC_{diff} \tag{4}$$

wobei

D_1, D_2 zwei zu vergleichende DC-Deskriptoren

W_1, W_2 Gewichtungsfaktoren

SC_{diff} absolute Differenz der beiden räumlichen Kohärenzen

DC_{diff} Ergebnis der Distanzfunktion (3)

$$DC^2_{diff}(D_1, D_2) = \sum_{i=1}^{N_1} p^2_{1i} + \sum_{j=1}^{N_2} p^2_{2i} - \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} 2a_{1i,2j}\, p_{1i}\, p_{2j} \tag{5}$$

wobei

$a_{k,l}$ Ähnlichkeitskoeffizient zweier Farben c_k und c_l mit der Bedingung:

$$a_{k,l} = \begin{cases} 1 - d_{k,l}/d_{max} & d_{k,l} \leq T_d \\ 0 & d_{k,l} > T_d \end{cases} \tag{6}$$

wobei

$d_{k,l}$ Euklidische Distanz zwischen den beiden Farben c_k und c_l

T_d ein Schwellenwert, bis zu jenem die zwei Farben als ähnlich angesehen werden

Je niedriger diese berechnete Distanz ist, desto mehr ähneln sich die beiden Bilder in ihren dominanten Farben.

2.1.2 Form/Gestalt

Wie schon zuvor angesprochen, unterliegt der Vergleich von Farb-Features gewissen Einschränkungen. So werden dem Benutzer, der eine Suchanfrage an die Multimedia-Datenbank stellt, möglicherweise von der Farbverteilung sehr ähnliche Ergebnisse geliefert, diese können jedoch aufgrund anderer Merkmale unerwünscht sein.
Ein weiteres wichtiges Merkmal im Image Information Retrieval stellt die Form oder die Gestalt von Objekten im Bild dar. So ist es nicht unwesentlich die Suche nach einem Sonnenuntergang am Strand nicht nur durch mögliche Farbintervalle zu definieren, sondern auch über gesuchte (typische) Formen (shapes) oder Bereiche.
Diesem shape-matching, bei dem ausgehend von einer Suchform (Skizze, Bild) ähnliche Formen gesucht sind, steht die Aufgabe der shape-representation/

-recognition gegenüber, bei der versucht wird ein Objekt anhand dessen geometrischen Form zu erkennen und einzuordnen.

Bevor noch diverse Shape-Features aus Bilddaten extrahiert werden, finden üblicherweise einige vorbereitende Schritte statt, die das Bildmaterial und die Konturen darin optimieren - von störendem Rauschen oder unsauberen Linien befreien.
Die wichtigsten Schritte dabei sind:

1. Binärisierung und Rausch-Reduzierung
   um ein Graustufen-Bild von Artefakten und Rauschpixeln zu befreien, wird mit einem einfachen Schwellenwert-Algorithmus das Bild in ein Binärbild gewandelt (zwei Farben: schwarz/weiß; ab einem gewissen Grautonwert wird das Pixel zu einem Weißen - darunter bleibt es schwarz). Dadurch liefern Shape-Features saubere Konturen.
   Weiters werden isolierte Pixel oder kleine Regionen eliminiert um die Shape-Analyse auf das Wesentliche zu beschränken.

2. Pixelergänzung
   In der Bildvorbereitung wird aber Information nicht nur gebündelt; mit Hilfe von connection techniques [20] wird automatisiert versucht, fehlende Pixel auf der Kontur zu ergänzen um einen lückenlosen Kantenzug zu erhalten.

3. Kantenglättung
   mittels 8-connectivity contour tracing Techniken [25], welche aufgrund festgelegter Regeln und Schwellenwerte einzelne Pixel einer Kante zuordnen oder davon ausschließen, wird zum Schluss eine Glättung der Kante erreicht. Dieser Schritt ist vor allem auch für die Vergleichbarkeit mit anderen Bildern vorteilhaft, da das Objekt auf seine Grundform reduziert wird und Details ausgeblendet bleiben.

Die eigentlichen Shape-Features, die eine Kontur analysieren und beschreiben, lassen sich im Wesentlichen in zwei Kategorien einteilen [21]:

- Rand-/Außenlinie-basierte Methoden beschränken sich bei der Suche und Beschreibung von Objektformen ausschließlich auf deren Kanten und Grenzen. Einige dieser Methoden bzw. Features sind chain codes, perimeter, fourier transformation, etc.
- Regionen-basierte Methoden hingegen beziehen zusätzlich die Information innerhalb dieser Grenzen mit ein und lassen so auch Rückschlüsse auf überlappende Objekte, Löcher oder den generellen Bildaufbau zu. Vertreter hierbei sind Moment-Invarianten.

Die Komplexität dieser Descriptoren ist sehr unterschiedlich; sie reicht vom bloßen Abzählen der Pixel einer Kante (Kantenlänge; auch Perimeter genannt) bis zu Zeit-Frequenz-Transformationen wie Wavelet.

Kanten-basierte Features: Chain coded string Die Außenlinie eines Objekts kann mit oder gegen den Uhrzeiger verfolgt werden. Der Chain Code dokumentiert diesen Trace für jedes Pixel. Pro Pixel stehen acht Codes zur Verfügung, welche für die Richtung des nächsten Pixels auf der Kante stehen (siehe Illustration 3).
Alternative Varianten dieses chain codes reduzieren die Anzahl der Codes durch andere Verschlüsselung der Richtungen (wie zum Beispiel Reduced Chain Codes oder Derivative Chain Codes).

Die Aufzeichnung/Dokumentation des Verlaufs einer zusammenhängenden Kante erlaubt es Vergleiche zwischen mehreren vorliegenden Objekten bzw. dessen Chain Codes mittels so genannter String-Distanz-Berechnungen anzustellen.

Liegen zwei Chain Codes A und B von Kantenverläufen vor

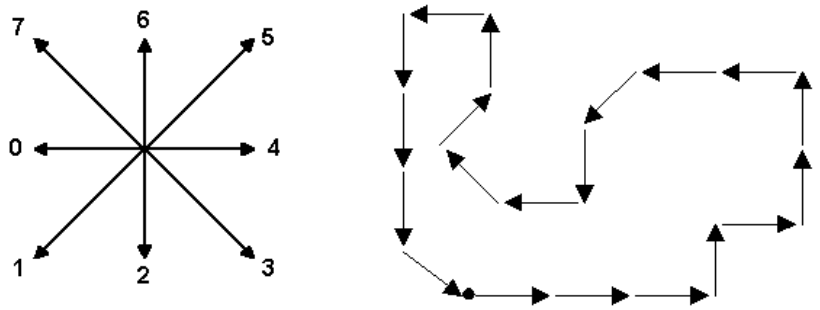

Abbildung 3: Chain Code: 3x3 Grid mit Richtungscodes und Beispiel-Kante

$A = a_1, a_2, ..., a_n$
$B = b_1, b_2, ..., b_m$

und sind s_k, i_k, d_k das Ersetzen, Einfügen und Entfernen von Werten um String A in B zu transformieren und weiters p, q, r die gewichteten Kosten dieser Operationen, so ist das String-Distanz-Maß - in diesem Beispiel die weighted Levensthein distance (WLD) wie folgt definiert:

$$D_{WLD} = min_k(s_k p + i_k q + d_k r) \tag{7}$$

Die zugrunde liegende Levensthein Distanz (DL) ist ein Maß für den Unterschied zwischen zwei Zeichenketten. Sie belastet mit jeder notwendigen Operation (Einfügen/Löschen/ Ersetzen) das Endergebnis mit +1. Je ähnlicher sich also zwei Zeichenketten sind, umso kleiner die Endzahl. Bei der gewichteten LD können die unterschiedlichen Operationen unterschiedlich gewichtet werden.
Alternativen zur WLD sind das nonlinear elastic matching und die extended distance [4].

Kantenbasierte Features: Fourier Descriptoren Konturen in einem Bild können als periodische Funktionen beschrieben und mittels Fourierreihen dargestellt werden. Mit Hilfe der diskreten Fourier Transformation (DFT; [40]) werden die Konturen vom Orts- in den Frequenzraum transformiert.

$$\tilde{f}(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t}\, dt \tag{8}$$

Bereits die niedrigen Frequenzen geben dabei Aufschluss über die Grundform und -beschaffenheit der Konturen. Mit der Frequenzhöhe steigen auch die Details einer Kante; werden beim Image Retrieval jedoch meist ignoriert, da deren Berechnung im Verhältnis zum Nutzen zu viel Zeit kostet.

Region-based Features: Moment Descriptoren Momente sind statistische Eigenschaften einer Form und können auf binäre oder Graustufen-Bilder angewandt werden. Es gibt eine Vielzahl an verschiedenen Momenten; im Prinzip sind sie jedoch immer gewichtete Mittelwerte aus den Helligkeitswerten der einzelnen Pixel. Dabei kommt es stark auf die Fragestellung an, welche Eigenschaften eines Bildes/Objektes man sehen will und welche Momente daher zur Anwendung kommen.
Generell lassen sie sich in folgende Kategorien einteilen [34]:

- **nicht zentrierte Momente** beschreiben recht einfache Bildeigenschaften wie Fläche und Schwerpunkt
- **zentrale Momente** sind invariant bezüglich Translation (Parallelverschiebung) und beschreiben ausgehend vom Masseschwerpunkt eines Objekts dessen Ausrichtung.
- **rotationsinvariante Momente** sind sowohl gegen Translation, Skalierung und Rotation invariant. Schon 1962 beschrieb Ming-Kuei Hu in [11] die heute häufigst verwendeten Sätze an invarianten Momenten.

2.1.3 Textur/Muster

Auch die Textur eines Bildobjekts kann sehr gut bei der Bildanalyse und -vergleich verwendet werden. Als Textur wird die optische Wahrnehmung von Oberflächenstrukturen eines Objektes verstanden.

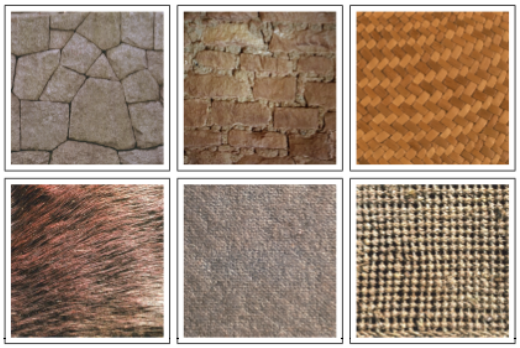

Abbildung 4: Beispiele an Oberflächen-Texturen

Ein großer Vorteil beim Miteinbeziehen von Texturbeschreibungen in die Mustererkennung von CBIR-Systemen (content based image retrieval) ist die Tatsache, dass es sehr viele charakteristische Texturen in der Umwelt gibt, die sich mit Hilfe geeigneter Deskriptoren beschreiben lassen und so eine sehr detaillierte Suche nach Bildern von bestimmten Objekten ermöglichen.

Beispiele für typische Texturen sind die Oberflächen von Gesteinen, Holz, Bakterien und menschliche Körperteile wie Iris und Fingerabdruck.

Nach [30] lassen sich texturanalytische Verfahren in drei Gruppen teilen:

- deterministische Verfahren beruhen auf der „Ermittlung von Texturprimitiven und ihrer Verteilung“ (Palm 2003, S.21 [24]). Die dafür notwendige Gleichmäßigkeit findet man jedoch allenfalls in künstlich generierten Texturen

- statistische Verfahren ziehen zur Charakterisierung von Texturen statistische Verteilungen heran (z.B. Cooccurrence-Matrizen [2])
- signaltheoretische Verfahren „zielen auf die Periodizität einer Struktur ab, die im Ortsfrequenzraum adäquat beschrieben wird“ [24] (z.B. Fourier Transformation, Gabor-Filter)

2.2 Video Information Retrieval

Da Videos immer aus Bildsequenzen bestehen, ist es theoretisch und praktisch möglich, sämtliche Image Feature Extraktoren auch auf einzelne Bilder eines Videos anzuwenden. Was jedoch bei Videos hinzukommt ist die zeitliche Komponente; und diese wiederrum spiegelt sich einerseits im Ton andererseits in der Bewegung der einzelnen Objekte im Bild wieder. Daher liegt das Hauptaugenmerk beim Video Information Retrieval in der Analyse der Bewegung im Bild und Segementierung einzelner Objekte über alle Einzelbilder hinweg.

2.2.1 Motion Detection/Estimation

Eine der wichtigsten Informationen, die man aus Videosequenzen extrahieren kann, ist die Bewegung im Bild. Sie kann über alle Frames hinweg analysiert (Detection) oder geschätzt werden (Estimation), was beispielsweise für die Kollisionsvermeidung bei Robotern oder Fahrzeugen [39], aber auch zur Kompression von Videomaterial [14] eingesetzt werden kann.
Die Information der Bewegung ist in vielerlei Hinsicht interessant: Sie gibt dem analysierenden System Auskunft über sich bewegende Objekte im Bild, was später für die Verfolgung und Segmentierung dergleichen verwendet werden kann.
Videokompressionsverfahren wie MPEG [9] versuchen unter anderem dadurch geringere Dateigrößen zu erreichen, indem nicht die volle Information einer Bildsequenz gespeichert wird, sondern gewisse Bereiche zur Laufzeit berechnet werden - dabei kommen ebenfalls motion estimation Verfahren zum Einsatz, welche die Bewegung eines jeden Blocks zwischen zwei Anker-Bildern während der Wiedergabe schätzen und entsprechend wiedergeben. Dadurch wird Speicherplatz eingespart, da gewisse Informationen zur Laufzeit aus den persistent Gespeicherten berechnet werden. Voraussetzung dafür ist natürlich die soft- bzw. hardwareseitige Unterstützung dieser Dekompressionsverfahren. So muss jedes Gerät, das Video-DVDs wiedergeben soll, die

Dekompression von MPEG2-komprimiertem Videomaterial beherrschen.
Die Techniken der Motion Detection können in zwei Hauptklassen geteilt werden:

- feature-based
  Grundlage für die feature-basierte Bewegungserkennung sind die - bereits bekannten - Objektfeatures Farbe, Form und Textur. Sie werden in sogenannten Block Matching Algorithmen dazu verwendet, Objekte im Bild blockweise von einem zum nächsten Frame zu verfolgen und so die Aufzeichnung eines Bewegungsvektors zu ermöglichen.

- motion-based
  Verfahren wie die Optical Flow Technique vergleichen in Bildsequenzen die Bewegung zwischen statischen und sich bewegenden Objekten; das Resultat sind so genannte Geschwindigkeitsvektoren (horizontal und vertikal), welche Grundlage für Bewegungsschätzung, -erkennung und Segmentierung sind.

Blockweise Methoden Zu den populärsten feature-basierten Bewegungserkennungs-Techniken gehören die Block-Matching Methoden [8].
Wie der Name schon vermuten lässt, teilen diese Algorithmen jeden Videoframe in Blöcke - meist von der Größe 8 x 8 Pixel (Farbe) oder 16 x 16 Pixel (S/W).
Ausgehend von einem Referenzframe wird nun für jeden Block über eine Fehler-/Kostenfunktion der Block im Zielframe ermittelt, der dem Ursprünglichen am ähnlichsten ist.
Die Fehlerfunktion kann dabei beliebig mächtig formuliert sein. Sie kann bloße Farbhistogramm-Vergleiche anstellen oder zusätzlich die Blöcke aufgrund Form und Textur mit einander vergleichen. Der Zielblock mit der geringsten Abweichung gibt Auskunft über die wahrscheinlichste neue Position des Ursprungblocks; und somit dessen Bewegung.

Um Rechenzeit einzusparen verwenden fast alle block-matching Algorithmen definierte Suchbereiche (z.B. 64 x 64 Pixel) um nicht über den gesamten Frame hinweg suchen zu müssen (siehe Abbildung). Diese Art der Abbruchbedingung muss aber natürlich vorsichtig und in Abhängigkeit der zu erwartenden Situationen definiert sein; denn ist der Suchbereich zu klein für die vorliegenden Bewegungen im Bild, führt die Suche nach dem Target-Block zu keinem oder einem falschem Ergebnis. Zu berücksichtigen ist hier auch die Framerate pro Sekunde - liegt vor allem Videomaterial mit einer sehr niedrigen Framerate vor, so muss davon ausgegangen werden, dass die Bewegungen zwischen zwei Frames größer (als bei hoher Framerate) sind. Der Suchradius sollte daher entsprechend groß veranschlagt werden.

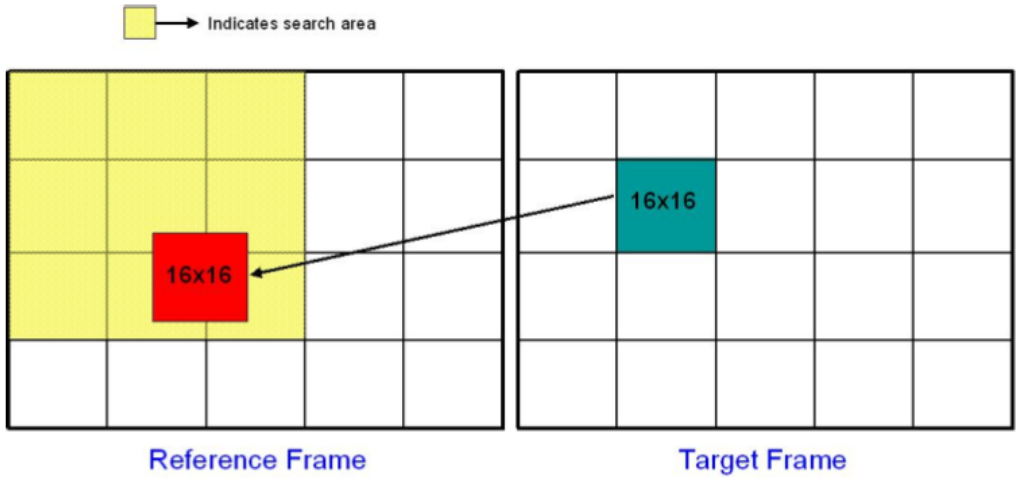

Abbildung 5: Block-Matching: Suche nach dem Block mit der höchsten Ähnlichkeit innerhalb der Suchregion

Der Erfolg beim Einsatz eines block-matching-Verfahrens ist also von mehreren Einflussfaktoren (Blockgröße, Suchregion, verwendete Features,...) abhängig, sodass ein breites Vorwissen über das zu analysierende Videomaterial ein entscheidender Vorteil bei der Entwicklung eines entsprechenden Systems ist.

Die wichtigsten Varianten der block-matching Verfahren sind

- exhaustive/full search
  unabhängig von einer Suchregion wird jeder Block im Targetframe mit jedem im Referenzframe verglichen und analysiert. Die Kostenfunktion ermittelt für jeden Vergleich die Differenz und beim Minimum wird das matching angenommen. Daraus kann für diesen und alle weiteren Blöcke der Bewegungsvektor ermittelt werden. Großer Nachteil dieser Methode ist zweifelsfrei die Rechenzeit, die dafür aufgewendet werden muss.

- step search
  die Suche wird schrittweise vollzogen: Ein gewählter Suchradius (in Pixel) wird mit jedem Schritt ausgeweitet; solange bis ein Ergebnis vorliegt. Üblicherweise werden aber maximal drei bis vier Suchschritte durchlaufen bis die Suche abbricht.
  Nach den Erkenntnissen von Jianhua Lu und Ming L. Liou [16] ist der three-step search Algorithmus (TSS) zu Recht die populärste Variante der block-matching Verfahren. Der Algorithmus arbeitet unter der Annahme, dass die Fehlerrate (des block-matching) monoton steigt, während sich der Vergleichspunkt vom globalen Fehlerminimum entfernt. Daher beschränkt er sich auf eine kleinere Suchbereichs-Größe und spart somit Laufzeit und Ressourcen.
  Eine Reihe an Modifikationen und Varianten der TSS versprechen noch geringere Laufzeiten und Komplexitäten bei gleichbleibender Qualität der Suchergebnisse. Erreicht wird dies durch Art und Umfang des Such-/Vergleichsschritts.
  Die wichtigsten Vertreter dabei sind
  - NTSS (new three-step search) [15]
  - FTSS (fast three-step search)[27]
  - SES (simple and efficent search)[16]

- 2D logarithmic search
  diese Verfeinerung der schrittweisen Suche sieht eine progressive Reduzierung des Suchgebiets (um die Hälfte) nach jeder Iteration vor; was wiederum zur Beschleunigung des Prozesses beiträgt.

In der Praxis werden meist mehrere Varianten mit einander kombiniert oder nacheinander angewandt. So kann der beste Kompromiss zwischen Laufzeit und Qualität des Resultats erzielt werden. Es darf jedoch nicht oberstes Ziel sein, die Laufzeit zu beeinflussen. Je kleiner der Suchbereich gewählt wird, umso größer auch die Gefahr, dass kein matching stattfindet.

Der Optical Flow stellt den bekanntesten Vertreter der motion-based Techniken dar und beschreibt - über jedes Bild - annähernd dessen Bewegungsfeld. Das Verfahren zur Bestimmung des OF wurde erstmals 1980 am MIT von Berthold Horn und Brian Schunck vorgestellt [10] und seither weiterentwickelt (z.B. von Lucas & Kanade [17]).

Als optical flow wird die sichtbare Bewegung von Helligkeitsmustern in einer Bildsequenz verstanden (konstante Helligkeit der Pixel in den beiden Frames vorausgesetzt), welche mittels entsprechender OF-Verfahren analysiert und für die weitere Verarbeitung verwendet werden kann [22].

Verfahren: durch die Identifikation von Helligkeitsmustern (Grauwert- Gradienten) in einem Bild/Frame wird differentiell für jedes Pixel (oder alternativ blockweise) der Motionvektor, der die Positionsveränderung des gleichen Pixels/-blocks im nächsten Frame beschreibt, bestimmt. Eine Kostenfunktion entscheidet - bei der Suche nach dem entsprechenden Pixel/-block im nächsten Frame - über die Übereinstimmung und speichert die Bewegungsinformation; hin zur neuen Position. Das alles geschieht jedoch unter der generellen Annahme, dass jeder Bildpunkt über alle Einzelbilder den selben Grauwert behält. Dass dies eine äußerst limitierende Einschränkung für reale Szenen bedeutet, muss bei der Auswertung der Ergebnisse berücksichtigt werden. Der Motionvektor besteht dabei aus zwei Komponenten, der horizontalen und vertikalen Verschiebung im Koordinatensystem. Die Auswertung dieses optischen Flusses, der mit Pfeilen unterschiedlicher Länge/Richtung als solcher visualisiert werden kann, gibt weiters Informationen über zusammenhängende Regionen (Objekte) im Bild, deren Anordnung im Raum und deren Bewegungsabläufe.

Der Algorithmus von Lucas & Kanade [17], der im Allgemeinen für MPEG Videosequenzen optimiert ist, beinhaltet im Wesentlichen drei Schritte:

1. Auswahl, Aufbereitung und Initialisierung
   Der erste Schritt umfasst die Auswahl der beiden zu vergleichenden Videoframes und deren Konvertierung zu Graustufen-Bildern, da für die Ermittlung des OF keine Farbinformationen gewünscht sind. Weiters werden sämtliche für die Berechnung notwendigen aber auch optionalen Parameter (wie die Größe der Suchregion oder die Anzahl der gauß-Pyramiden-Level) eingelesen und der Prozess initialisiert.

2. Berechnung der Ableitungen
   Zur Lösung der OF-Gleichung muss zuvor die partielle Ableitung der Bildhelligkeit (unter Berücksichtigung der X/Y-Achsen und der Zeit) berechnet werden. Dazu wird eine zwei-dimensionale Biegung auf die beiden Videoframes angewandt. Als Ergebnis liegen die beiden Matrizen A und b vor, welche in die OF-Gleichung eingesetzt werden können.

3. Lösen der OF-Gleichung
   Lucas & Kanade verwenden zur Lösung der OF-Gleichung

$$A\overrightarrow{u} = b \tag{9}$$

   die mathematische Methode der kleinsten Quadrate.

   Ausgehend vom vorliegenden überbestimmten System (anhand einer drei-dimensionalen Bewegung)

$$\begin{bmatrix} I_{x_1} & I_{y_1} & I_{\approx_1} \\ I_{x_2} & I_{y_2} & I_{\approx_2} \\ \dots & \dots & \dots \\ I_{x_n} & I_{y_n} & I_{\approx_n} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_\approx \end{bmatrix} = \begin{bmatrix} -I_{t_1} \\ -I_{t_2} \\ \dots \\ -I_{t_n} \end{bmatrix} \tag{10}$$

   können die drei gesuchten Flussvariablen (durch Berechnung der Ab-

leitungen) gelöst werden:

$$\begin{bmatrix} V_x \\ V_y \\ V_{\approx} \end{bmatrix} = \begin{bmatrix} \Sigma I_{x_i}^2 & \Sigma I_{x_i} I_{y_i} & \Sigma I_{x_i} I_{\approx_i} \\ \Sigma I_{x_i} I_{y_i} & \Sigma I_{y_i}^2 & \Sigma I_{y_i} I_{\approx_i} \\ \Sigma I_{x_i} I_{\approx_i} & \Sigma I_{y_i} I_{\approx_i} & \Sigma I_{\approx_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\Sigma I_{x_i} I_{t_i} \\ -\Sigma I_{y_i} I_{t_i} \\ -\Sigma I_{\approx_i} I_{t_i} \end{bmatrix} \tag{11}$$

V_n ist dabei der Bewegungsvektor auf der n-Achse.

Anwendungsgebiete kennt der Optical Flow viele; bekannte Beispiele sind die Kollisionsvermeidung (Navigation von Robotern/Autopiloten), die automatische Verfolgung/Segmentierung von bewegten Objekten (Verkehrsüberwachung) und die Gestik-/Mimikerkennung im menschl. Gesicht.

Einschränkungen: Wie schon erwähnt, ist das Ergebnis jedoch nur eine näherungsweise Schätzung der tatsächlichen Bewegung (in der realen Welt). Denn stellt man sich zum Beispiel eine sich drehende Glaskugel vor, so kann diese Bewegung durch ein OF-Verfahren nicht erkannt werden, da praktisch keine Farb-/Helligkeitswechsel im betrachteten Bild stattfinden - es scheint statisch zu sein. Ein weiteres Problem, vor dem auch diese Arbeit steht, ist die Tatsache, dass dieses Verfahren zwischen einer Bewegung der betrachteten Objekte und der Kamerabewegung (Pan/Tilt/Zoom/Jitter) *nicht* unterscheiden kann. Für das Verfahren findet in beiden Fällen eine Veränderung des Grauwertbildes statt und sucht je Pixel den Standort im nächsten Frame.

Herausforderungen für Bewegungserkennungs-Techniken Viele Experimente mit unterschiedlichen Varianten werden in der Literatur beschrieben, denen aber allen gemein ist, dass sie unter „Laborbedingungen“ - also selektiertem Bildmaterial - stattfinden. Störfaktoren, mit denen man sich bei der Entwicklung von Endprodukten auseinandersetzen muss, sind unter anderem:

- Geschwindigkeit

Optical Flow Algorithmen gehen von einer relativ kleinen Bewegung zwischen zwei Frames aus. Wird daher die Framerate (Bilder pro Zeiteinheit) zu klein gewählt, stoßen diese Verfahren an ihre Grenzen und liefern keine aussagekräftigen Resultate. Je höher die Framerate ist, umso öfter wurde eine Szene in Bildern festgehalten, desto kleiner ist daher die Durchschnittsbewegung von Frame zu Frame.

OF-Verfahren bieten dafür aber auch integrierte Lösungen an. So hat sich neben dem Erhöhen der Framerate ein weiteres Konzept bei der Optimierung der OF-Berechnung etabliert, das zwei Überlegungen/Vorteile kombiniert: Verkleinert man das Bild als Ganzes, werden auch die Bewegungen darin kleiner und für OF-Verfahren erkennbar. Weiters kann eine Optimierung der Berechnung des OF durch Reduktion des Bildrauschens erreicht werden. Schlechte Lichtverhältnisse, Reflexionen oder zu starke Kompression rufen häufig störendes Bildrauschen hervor, welches die Ergebnisse einer OF-Analyse verfälschen.

Das Konzept der Gauß-Pyramide sieht daher die Anwendung eines blur-Filters auf die zu analysierenden Bilder vor. Dabei werden die Details eines Bildes Schritt für Schritt reduziert und somit geglättet. Abbildung 6 veranschaulicht die Verdichtung des Bildmaterials mit jedem Level - der Bildinhalt wird mit jedem Schritt um die Hälfte verkleinert und dessen Detaillevel damit verringert. Neben der Verringerung des Detailgrades kommt die Verkleinerung der Bewegungen der OF-Analyse entgegen. Die Anzahl der Stufen/Schritte ist dabei wieder an die jeweilig zu erwartenden Szenen anzupassen und auszurichten.

- Dynamik des Hintergrundes
  Regen, fallende Blätter oder große Menschenansammlungen im Hintergrund stellen eine weitere große Herausforderung für die Bewegungser-

kennung dar, da gerade Verfahren wie der Optical Flow versucht statische von dynamischen Bildbereichen zu unterscheiden, ist es oft erforderlich relevante Bereiche zu segmentieren (siehe auch Kapitel 2.2.3) um irrelevante Bewegung auszublenden.

- Änderungen in der Helligkeit
  sich ändernde Lichtverhältnisse oder Kamerabewegungen können einen eigentlich statischen Hintergrund für Motion Detection Algorithmen bewegt erscheinen lassen und so deren Ergebnisse deutlich beeinflussen. Ein Algorithmus muss daher robust genug sein, solche Störfaktoren zu ignorieren und die eigentliche Bewegung (des Vordergrunds) analysieren.

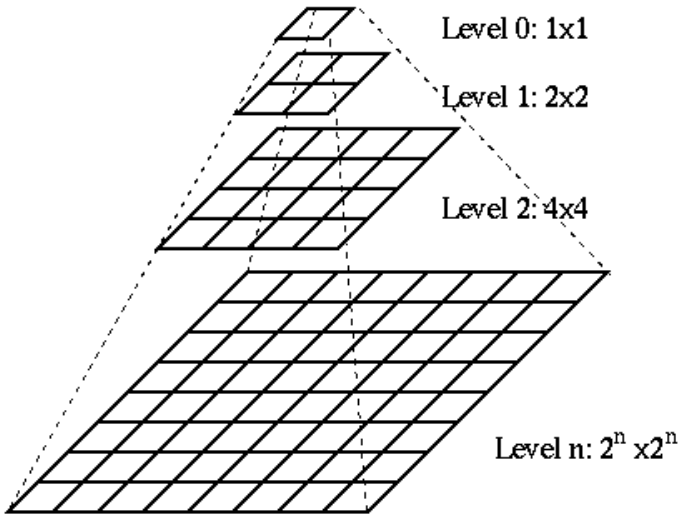

Abbildung 6: Gauß-Pyramide: Reduktion der Details und Verkleinerung des Bildes mittels Gausschem Filter (Darstellung der Pyramiden-Level)

2.2.2 Image Stabilization

Wird ein Video mit einer (Hand-)Kamera ohne fixer Verbindung zum Boden (also ohne Stativ bzw. ähnlichem) aufgenommen, kommt es meist zu ungewollten Verwackelungen. Bei vielen Anwendungen können diese den Analyse-Prozess deutlich stören bzw. die Ergebnisse verfälschen.
Will man also den genauen Bewegungsverlauf eines Objekts analysieren, müssen allfällige Verwackelungen minimiert/ausgeschlossen werden. Hier bedarf es einer Methode das Objekt „stabil“ im Bild zu verankern. Die meisten modernen Digitalkameras (Foto und Video) bieten bereits eine Lösung zur Bildstabilisierung an. Zum einen kann dies eine reine Softwarelösung sein, zum anderen gleichen bereits viele Objektive Erschütterungen mechanisch aus. Bei zweiterem sind es Bewegungs- und Beschleunigungssensoren, die dem Objektiv mitteilen, in welchem Ausmaß eine Ausgleichsbewegung (mit Hilfe kleinster Motoren) stattfinden muss, um die zitternden Bewegungen der Hände auszugleichen.

Die Alternative dazu sind rein software-basierende Verfahren. Generelle Überlegung bei diesen Verfahren ist die Tatsache, dass das Videobild als Ganzes über die Zeit ruhig bleibt (nämlich vorgegeben aufgrund Format und Auflösung der Kamera) und nur dessen Inhalt sich ändert, wackelt.
Man kann aber natürlich versuchen das umzukehren und gewisse Objekte im Bild an deren Ausgangslage zu binden. Dazu muss das Bild als Ganzes im gleichen Ausmaß bewegt werden, wie Verwackelungen usw. auftreten. Erreicht wird das durch frameweise Ermittlung eines Verschiebungsvektors (annähernd gleich der Bewegung/Verwackelung, die vom Kameramann verursacht wird), der die Lage eines Pixels im darauffolgenden Frame bestimmt. Diese weicht natürlich von der ursprünglichen Position ab und berücksichtigt nun für jeden Pixel die auszugleichende Verwackelung.
Ein besonderer Umstand muss jedoch beachtet werden; aufgrund größerer Verschiebungen von Pixeln kommt es natürlich - vor allem an den Randberei-

chen - immer wieder zu einem Informationsmangel und somit schwarzen/leeren Pixeln. Sollen diese am Ende nicht sichtbar sein, sollte der Bildausschnitt so verkleinert werden, dass Bereiche, die zumindest einmal während des Clips schwarz erscheinen, weggeschnitten werden.
Als Ergebnis erhält man ein - die Hintergrundbewegung betreffend - annähernd ruhiges Bild.

2.2.3 Region/Foreground Segmentation

Um wirklich Personenvergleiche bzw. -unterscheidungen anstellen zu können, ist es erforderlich die Person/Gestalt vom restlichen (uninteressanten) Hintergrund (zB. Schnee) zu trennen und nur davon weitere Berechnungen anzustellen.
Doch wie kann dies geschehen? Verschiedene Ansätze und Methoden haben sich mittlerweile entwickelt, deren Einsatz stark von den zu erwartenden Szenen, die segmentiert werden sollen, abhängt.

Im Wesentlichen bestehen zwei Ansätze zur Segmentierung eines Bildes, wobei die Kombination der beiden (für komplexe Aufgaben) sicher die vielversprechendste (aber auch aufwändigste) Lösung darstellt.

color-based Segmentierung [13] basiert vor allem auf der vorhanden Farbinformation jedes Pixels. Die generelle Funktionsweise bei den meisten Ansätzen geht von der Ermittlung so genannter Cluster (~Objekte/Regionen im Bild) aus, denen benachbarte Pixel - aufgrund eines Distanzmaßes (Farbe, Intensität, Textur,...) - zugeordnet werden. So wächst jeder Cluster bis am Ende, jedes Pixel einer Region zugeordnet ist und somit eine überschaubare Anzahl an Regionen identifiziert wurde. Das Ergebnis kann dann noch weiter verfeinert bzw. reduziert werden.
Das Problem für unseren Anwendungsbereich ist hier jedoch die fehlende Information, welche Region die Person/Gestalt beinhaltet.

Hier kommt uns das zurückgelieferte Ergebnis des MPEG-7 Dominant Color Features zugute. Das Verfahren generiert für jede identifizierte Dominant Color eine Maske (binäres Bild in dem die Pixel der DC weiß alle anderen schwarz sind), die in unserem Anwendungsgebiet gute Segmentierungsergebnisse liefert (siehe Kapitel 3.2).
Diese Maske kann dann in weiterer Folge über das Originalbild gelegt und (nun lokale) Analysen angestellt werden, da nur noch relevante Pixel (nämlich jene, die den Snowboarder bilden) vorliegen.

motion-based Der zweite viel versprechende Ansatz ist jener der motion-based Segmentation. Will man ein Objekt (z.B. Person), das sich vor einem statischen (!) Hintergrund bewegt segmentieren, so ist dies durch eine frameweise Ermittlung des Hintergrundes möglich [26]. Mit jedem Frame werden jene (Hintergrund-) Bereiche ermittelt, die vom sich bewegenden Objekt neu verdeckt bzw. sichtbar gemacht wurden. Kombiniert man diese Informationen über alle Frames hinweg, erhält man den (hoffentlich) vollständigen Hintergrund und kann im nächsten Schritt für jedes Frame diesen vom Vordergrund (= Objekt) trennen. Als Ergebnis erhält man eine Bildsequenz, die ausschließlich die Person bzw. das Objekt zeigt.

Vor allem in der Notwendigkeit Personen oder Objekte aus Videosequenzen entfernen zu können, begründete David B. Brown zu Beginn dieses Jahrzehnts [3] seine Motivation einen effizienten Algorithmus dafür zu entwickeln. Anhand dieses Verfahrens lassen sich die wichtigsten Schritte und Probleme gut darstellen:

1. Rauschunterdrückung
   Wie bei den meisten Information Extraction Verfahren sind die ersten Schritte eher vorbereitender Natur. So ist es laut [3] unerlässlich vor der eigentlichen Motion Estimation jene Pixel zu identifizieren und auszuschließen, deren Bewegung irrelevant bzw. störend sind. Empfohlen

wird dafür das Berechnen der mittleren quadratischen Fehler und deren Abweichungen. Mit Hilfe des dadurch ermittelten Schwellenwerts lassen sich unerwünschte Pixel(-regionen) vom weiteren Algorithmus ausschließen.

2. Motion Estimation
   Im nächsten Schritt werden für jedes Frame-Paar die Bewegungsvektoren mit Hilfe von entsprechenden Verfahren (siehe auch 2.2.1) ermittelt. Dabei sieht man sich natürlich den gleichen Herausforderungen gegenüber, wie bereits beschrieben - Suchfenstergröße und sonstige suchoptimierende Parameter müssen ausgewählt und evaluiert werden.

3. Motion Tracer
   Wie auch vorhin schon beschrieben, ist die grundlegende Annahme für den eigentlichen Segmentierungsschritt, dass es sich um eine Videosequenz handelt, in der sich ein oder mehrere Objekte vor einem *statischen* Hintergrund bewegen. Nur dann liefert folgendes Verfahren die erwünschten Ergebnisse.
   Die Aufgabe des Motion Tracer ist es, sämtliche Regionen zu identifizieren, die sich irgendwann, in irgendeinem Frame bewegt haben. Zentral dabei ist, dass diese Information auf die gesamte Videosequenz angewandt wird und nicht nur auf jene Frames, in denen die Bewegung stattgefunden hat. Das Erkennen von zusammenhängenden sich bewegenden Regionen ist ein rekursives Problem. Jeder Block in dem eine Bewegung stattgefunden hat, wird zu einem weiteren Ausgangspunkt für den nächsten Blockvergleich.

   Wie in Abbildung 7 dargestellt, gibt es dabei drei Varianten. Sie unterscheiden sich sowohl in der Tatsache was sich der Algorithmus als bewegte Regionen merkt (graue Bereiche) und welche benachbarten Blöcke in die weitere Analyse rekursiv einfließen (weiße Kreise).

der erste pixelorientierte Ansatz berücksichtigt zwar alle „berührten“

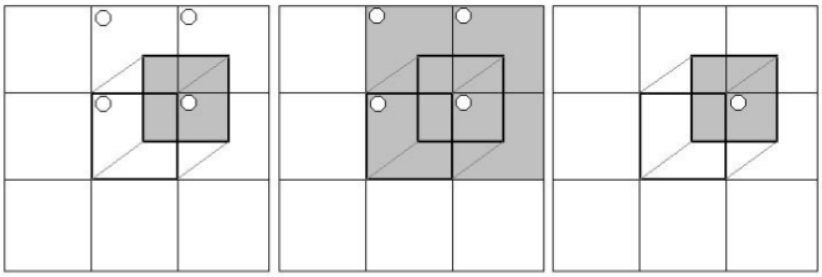

Abbildung 7: drei Varianten des Motion Trace Algorithmus im Bezug auf seine rekursive Arbeitsweise; dabei sind jene Bereiche grau gekennzeichnet, die der Algorithmus als (zumindest einmal) bewegt identifiziert hat

Blöcke in den nächsten Tracing-Schritten, merkt sich aber nur die sich tatsächlich bewegenden Pixel eines Blocks.

beim mittleren Ansatz sind ebenfalls alle von der Bewegung betroffenen Blöcke Ausgangspunkt für das weitere Tracing, es wird aber zusätzlich auch die gesamte betroffene Region als einmal bewegt vermerkt, was in den späteren Resultaten zwar zu keinem exakten enganliegendem Schnitt zwischen Vorder- und Hintergrund führt, aber die Wahrscheinlichkeit minimiert, dass Pixel innerhalb des Objekts ausgeblendet werden. Weiterer Vorteil ist der Performance-Gewinn; ein getracter Block muss kein zweites Mal analysiert werden.

beim dritten Verfahren werden wie beim ersten nur jene Pixel vermerkt, die sich tatsächlich bewegen. Abweichend jedoch findet nur jener Block mit der maximalen Überlappung den Weg in die weitere Tracing-Kette. Dieser Ansatz ist vor allem wegen seines Performance-Gewinns relevant.

4. Interpolation
   Ein letzter - optionaler - Schritt kommt nur dann zur Anwendung, wenn man sich zuvor für die rechenintensiven Methoden entschieden hat und

nun Rechenzeit durch Überspringen von Frames einsparen will. Die Analyse von Bewegung muss nicht unbedingt zwischen jedem einzelnen Frame stattfinden. Oft reicht es die Bewegung in Abständen von fünf Frames zu berechnen und danach eine Interpolation der Bewegung zwischen den - weiter entfernten - Frames anzuwenden.

Überlegung Die limitierende Einschränkung dieser Verfahren, dass ein statischer Hintergrund Voraussetzung ist, macht sie für das Vorhaben dieser Arbeit leider unbrauchbar. Der Kameramann hat den Snowboarder meist im Bild verfolgt und nicht die Kamera auf einen Hintergrundpunkt fixiert. Außerdem wurde bei den Dreharbeiten kein Stativ verwendet, weshalb der Hintergrund schon alleine wegen der Verwackelungen nicht statisch ist (siehe 2.2.2).

2.3 Evaluierung

Mit der Auswahl, Definition und Extraktion von Bild-/Video-Features einher, geht die Evaluierung deren Qualität, Ergebnis und Relevanz für das vorliegende Problem. Vor der Entwicklung eines CBIR Systems können nur theoretische Überlegungen bei der Auswahl der notwendigen Features einfließen und es besteht das Risiko, dass sie sich im Nachhinein für die vorliegende Fragestellung als unpassend erweisen.
Um diesen Evaluierungsschritt zu formalisieren und unterschiedliche Feature-Ergebnisse bewerten zu können, hat sich im Bereich des Multimedia Information Retrieval eine eigene Disziplin entwickelt, deren primäres Ziel es ist die Qualität des Ergebnisses in Relation zum eingesetzten Aufwand zu setzen.
So werden in der Literatur einige Maße und Kennzahlen angeboten, die bei der Bewertung von Features helfen sollen und sich grob in zwei Kategorien teilen lassen:

- **benutzerorientierte Kennzahlen/Systeme**
  versuchen sich auf die Präferenzen und Erwartungen eines einzelnen Benutzers zu beziehen. Kennzahlen dieser Kategorie sind daher nicht als statisch, endgültig oder fix zu betrachten, sondern erwarten immer in einem zweiten Schritt die Interaktion bzw. Feedback des Benutzers. Ein Parameter, der in diese Kennzahlen miteinflißt ist die Größe der relevanten Suchergebnisse, die im höchsten Maße subjektiv sein kann und somit von Benutzer zu Benutzer zu unterschiedlichen Ergebnissen führen kann (z.B. Suche nach besonders schönen Fotos bei Sonnenuntergang).
  Eines der bekanntesten Kennzahlensysteme ist Precision und Recall, welches die Güte von Treffermengen bei einer CBIR-Search wie folgt definiert [18]:

  Die Trefferquote ist die Wahrscheinlichkeit mit der ein relevantes Do-

kument (Bild, Video,...) gefunden wird:

$$recall = \frac{|relevanteDokumente \cap gefundeneDokumente|}{|relevanteDokumente|} \tag{12}$$

Die Genauigkeit ist die Wahrscheinlichkeit mit der ein gefundenes Dokument relvant ist.

$$precision = \frac{|relevanteDokumente \cap gefundeneDokumente|}{|gefundeneDokumente|} \tag{13}$$

Zusätzlich ist es möglich die Wahrscheinlichkeit zu berechnen, mit der dem Benutzer ein irrelevantes Dokument vorgelegt wird (Ausfall):

$$fallout = \frac{|gefundene - relevante|}{|alle - relevante|} \tag{14}$$

Das Problem dieser Kennzahlen in der Praxis ist, dass man nur selten die Zahl der relevanten Dokumente in der Datenbank kennt. Dafür ist entweder Wissen über das vorliegende Material notwendig (Heranziehen von Experten) oder man versucht aufgrund von Stichproben auf die Gesamtheit der Datenbasis Rückschlüsse anzustellen.
In der Literatur finden sich weiters kombinierte und abgeleitete Maße der zuvor genannten Kennzahlen. Das Effektivitätsmaß [38] zum Beispiel entspricht dem gewichteten (α) harmonischen Mittel aus Precision und Recall. Das Ergebnis liegt zwischen 0 (beste Effektivität) und 1 (schlechteste Effektivität).

$$E = 1 - \frac{1}{\alpha(\frac{1}{Precision}) + (1-\alpha)\frac{1}{Recall}} \tag{15}$$

- **systemorientierte Maße**
  geben eine globale Sicht auf eine benutzerunabhängige Beurteilung der Retrieval-Ergebnisse. Ein Beispiel dafür ist das Nützlichkeitsmaß von

Frei und Schäuble [7].

2.3.1 Relevance Feedback

Eine benutzerorientierte Möglichkeit Suchergebnisse zu verbessern und so die Precision zu erhöhen, ist die Funktionalität des Relevance Feedback. Darunter wird das Einbeziehen des Benutzers in den CBIR-Prozess verstanden, der damit versucht die semantische Lücke so gut wie möglich zu schließen.

Die semantische Lücke (semantic gap [36]) bezeichnet den Umstand, dass zu einer vollständigen Beschreibung von Objekten und deren Zuständen meist Kontextwissen vorhanden sein muss, ohne dem ein Objekt und dessen Zustand falsch oder gar nicht interpretiert werden kann. So ist es für den Menschen mit seinem Werkzeug der Sprache relativ einfach komplexe Zusammenhänge in der realen Welt zu formulieren. Wird jedoch versucht diese Information mit Hilfe weniger mächtiger Sprachkonstrukte - wie Programmiersprachen - abzubilden, stößt man heute noch an Grenzen und so entstehen Unterschiede zwischen der Situation in der realen Welt und der reproduzierten Abbildung in Form von Algorithmen und Datenpools.
Weiters können die mächtigsten CBIR-Verfahren/-Techniken und -Methoden; Features und Deskriptoren, nie an die menschliche Wahrnehmungskraft und -weise heranreichen. Sie liefern zwar sehr rasch - aufgrund messbarer Merkmale - passende Ergebnisse, doch bringt erst der Betrachter das Gesuchte und Gesehene in Kontext mit Gefühlen oder Details. Es ist zwar zum Beispiel möglich einem System beizubringen menschliche Freude aufgrund der Mimik zu erkennen, doch Feinheiten oder sogar das Vortäuschen freudiger Gefühlsregungen entziffert nur das menschliche Gehirn.
Der Prozess findet dabei iterativ - also in Schritten/Runden - statt. Das CBIR-System legt dem Benutzer - seinen Berechnungen nach - relevante Dokumente vor und gibt dem Benutzer die Möglichkeit zu definieren, welche davon besonders passend oder irrelevant sind. Mit jeder dieser Runden wird

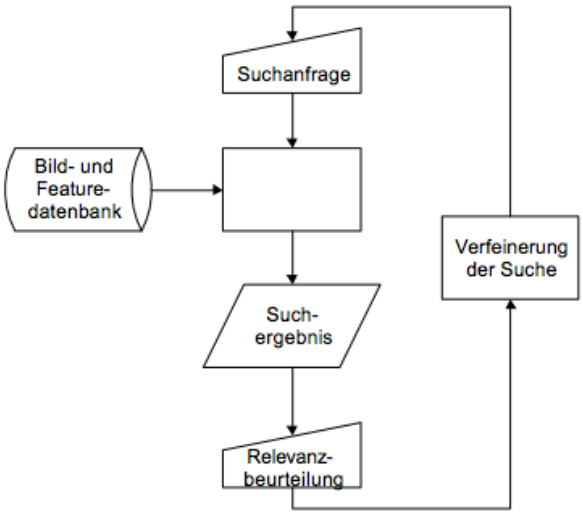

Abbildung 8: der iterative Prozessablauf des Relevance Feedback[6]

das Ergebnis immer relevanter, die Precision höher und die Ergebnismenge kleiner. Dadurch unterstützt der Benutzer das System mit seiner individuellen Wahrnehmung und Erwartung. Das System wiederum übernimmt mit seiner mächtigen Rechenleistung das aufwändige Berechnen, Vergleichen und Auswählen von relevanten Dokumenten. Der Erfolg liegt somit im Zusammenspiel der beiden.

Die Qualität und Effektivität des Relevance Feedback hängt jedoch von der letztendlichen Implementierung ab. So kann der Schwerpunkt zwischen einer Endbenutzer-freundlichen Feedback-Lösung - mit minimalistischer aber einfach zu bedienender und schneller Programmlogik - und einem mehrstufigen ev. sogar selbst lernendem (kernel-based) Refinement-Prozess, der dem Benutzer durch geschickte „Befragung“ und Analyse der Antworten bis zum gewünschten Ergebnis begleitet, liegen.

Fortgeschrittene Verfahren des Relevance Feedback, wie in [32], versuchen aus dem Feedback-Verhalten des Benutzers eine dynamische Gewichtung der relevanten Merkmale zu ermitteln, um so proaktiv auf die individuelle Herangehensweise des Benutzers einzugehen. Das Ziel dabei ist, dass der Benutzer nicht wissen muss, wie er dem System seine Suchfrage formulieren muss; er stellt zu Beginn eine Initial-Suchanfrage und durch sein gezieltes Feedback adaptiert das System die Suchanfrage mit jeder Iteration.

Das integrierte Relevance Feedback nach [32] wählt bereits aufgrund der gestellten Suchanfrage relevante Features aus, zerlegt die Suchanfrage danach und vergleicht diese mit den berechneten Objektfeatures aus der Multimedia-Datenbank (vgl. Abbildung 9). Dieser Ansatz ist insofern überzeugend, da bisherige Refinement-Verfahren davon ausgegangen sind, dass der Benutzer seine Erwartungen in ein computer-zentrisches Schema übersetzen kann.

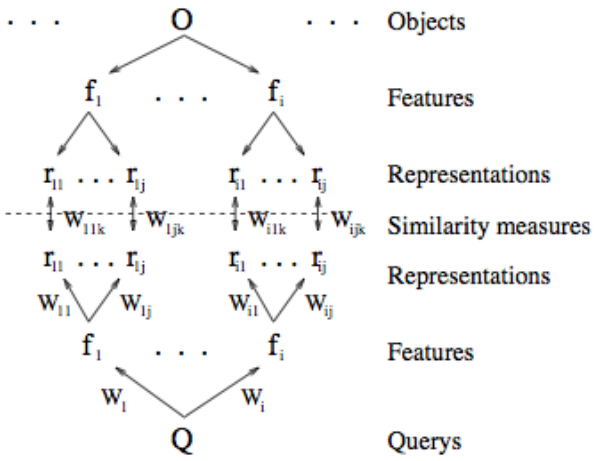

Abbildung 9: integriertes Relevance Feedback stellt den Objektfeatures die dynamisch ermittelten Queryfeatures gegenüber

2.4 Zusammenfassung

Zusammenfassend kann hier festgehalten werden, dass sich die Probleme, die bei den gesetzten Zielen dieser Arbeit zu lösen sind, durchaus in diversen Methoden und Techniken der VIR wiederspiegeln und Lösungen vorgeschlagen werden. Sämtliche Teilprobleme konnten bei der Recherche zumindest theoretisch gelöst werden und es erfordert nun einer Kombination und ggf. Erweiterung der gefundenen Techniken um den Anforderungen gerecht zu werden.
Ausgehend von der Problemdefinition sind vor allem folgende Themenbereiche für die Arbeit interessant und sollen in die Umsetzung miteinfließen:

- Optical Flow: Zur Bestimmung der Bewegung, die durch den Snowboarder vollzogen wird, ist die Extraktion des Optical Flow erforderlich und die gefundenen Lösungen liefern sehr gut verwertbare Ergebnisse.

- Color Feature: Sowohl zur Differenzierung einzelner Personen (aufgrund ihrer Kleidungsfarbe), als auch zur Trennung von Schnee und Snowboarder ist das Konzept rund um den Dominant Color Descriptor des MPEG-7 Standards sehr vielversprechend.

Die folgenden Kapitel dokumentieren die Experimente und deren Ergebnisse, die mit Hilfe der gefundenen Techniken erzielt wurden.

3 Umsetzung/Implementierung

Ausgehend von der Problemdefinition, wonach zwei Hauptprobleme zu bewältigen sind, wurden bestehende Ansätze auf ihre Relevanz hin untersucht und in einem Prototyp kombiniert.

Aufgrund

- der guten Unterstützung für sehr große, mehrdimensionale Datenmengen/-matrizen (wie sie bei Videodaten vorliegen; jedes Bild/Frame eines Videos wird als mehr-dimensionale Matrix der Farb-/Helligkeitswerte je Pixels digitalisiert),
- den weitgreifenden Integrationsmöglichkeiten fremdsprachiger Codeelemente (C, C++, Java...) sowie
- dem - in der Lehrveranstaltung bereits erlangtem - Wissen darüber

wurde für diese Aufgabe Matlab der Firma The MathWorks als Entwicklungsumgebung/ -werkzeug gewählt.

Nachfolgend soll nun der Lösungsweg und dessen Ergebnisse dargelegt werden.

3.1 Teilaufgabe: Klassifikation nach Schwungarten

Wie schon in der Problemdefinition beschrieben, ist *ein* Ziel der Arbeit, die automatisierte Klassifikation von Snowboard-Clips nach Schwungarten.

3.1.1 theoret. Überlegungen

Unter einem Schwung (nicht nur im Schnee) versteht man prinzipiell den Richtungswechsel eines Objekts, einer Linie oder Funktion, dessen Verlauf sprunghaft aber auch ausgedehnt sein kann.

Für die Klassifikation nach Schwungarten maßgeblich ausschlaggebend ist das Schwungintervall - also die Dauer, die zwischen zwei Schwüngen, vergeht. Beim Kurzschwung ist diese Zeitdauer - wie der Name schon vermuten lässt - eher kürzer (da versucht wird möglichst viele Schwünge auf kleinem Raum bzw. in kurzer Zeit zu fahren), beim (weit ausgedehnten) Carving-Schwung zum Teil sehr lange.
Diese Richtungswechsel können auf Grundlage des Optical Flow (siehe 2.2.1) bestimmt, deren Intervall berechnet und damit eine Klassifizierung getroffen werden.
Das Intervall ergibt sich aus der Anzahl der Frames, die zwischen zwei identifizierten Richtungswechsel liegen.

3.1.2 Vorarbeiten

Da das vorliegende Filmmaterial nicht mit einem Stativ, sondern per Hand gefilmt wurde, weißt es zum Teil starke Verwackelungen (jitter) auf, die die folgenden Berechnungen stören und das Ergebnis verfälschen würden.
Daher wurden alle Clips, nach zuvor gescheiterten Experimenten, mit einer speziellen Software (Image Stabilizer siehe Kapitel 2.2.2) „behandelt“, die dafür sorgt, dass Verwackelungen zumindest minimiert werden und somit ein ruhigeres Bild entsteht.
Bei der Umsetzung eines Systems für den Endanwender sollte entweder ein Stativ verwendet werden oder versucht werden den Stabilisierungsschritt (der natürlich Rechenzeit kostet) in den Gesamtprozess einzugliedern.

3.1.3 Implementierung

Grundlage für dieses Teilproblem ist - wie schon erwähnt - die Auswertung des Optical Flow der einzelnen Clips, da berechnet werden muss, wie lange die Zeitdauer ist, nach der der nächste Richtungswechsel stattfindet.
Dazu wurde für diese Implementierung die „Open Source 3D Vision Library“ - kurz openvis3d [23] - eingebunden und als Ausgangspunkt für die Kassifi-

kationsberechnungen festgelegt.
Openvis3d stellt verschiedene C++ Routinen für die Bild- und Videoverarbeitung zur Verfügung - darunter auch die Extraktion des Optical Flows (nach Abhijit S. Ogale).

Da bei der Videoaufnahme von Snowboardschülern meist vom Tal hinauf gefilmt wird und somit die Schwünge für den Betrachter von links nach rechts (und umgekehrt) zu erkennen sind, ist für die folgende Berechnung vor allem die horizontale Bewegung auf der x-Achse im Bild ausschlaggebend. Die vertikale (y-Achse) kann daher vernachlässigt werden.

Als Resultat der Optical Flow Extraktion eines Snowboard-Clips erhält man daher (unter anderem) die horizontalen Bewegungsvektoren jedes einzelnen Pixels in jedem Frame. Sehr einfach lässt sich diese Fülle an Information verdichten, indem *ein* globaler Bewegungsvektor (= Mittel aller Bewegungsvektoren eines Frames) ermittelt wird (Gesamtprozess siehe Abb. 10).
Nun liegt über die Länge des Clips hinweg ein Bewegungsvektor pro Frame vor, der die (horizontale) Grund-Bewegungsrichtung zu diesem Zeitpunkt angibt.
Richtungswechsel kennzeichnen sich dadurch, dass sich der Trend der Bewegung von einem zum anderen Frame ändert. Sehr gut erkennbar wird das anhand des Plots der globalen Bewegungsvektoren eines Beispiel-Clips (vgl. schwarze Punkte in Abbildung 11).
Da - trotz Bildstabilisierung - hier immer noch viele Verwackelungen vorhanden sind, kann nicht einfach jeder Peak (Spitze im Plot = lokales Maximum/Minimum = Richtungsänderung) als Schwung interpretiert werden. Es werden daher im nächsten Schritt jene Peaks identifiziert, deren Vorgänger einen gewissen Schwellenwert über- bzw. unterschreiten. Dadurch werden kleine Verwackelungen für die weitere Berechnung weitgehendst ignoriert. Als Schwellenwert für den Mindestabstand zweier Peaks wurde die Varianz

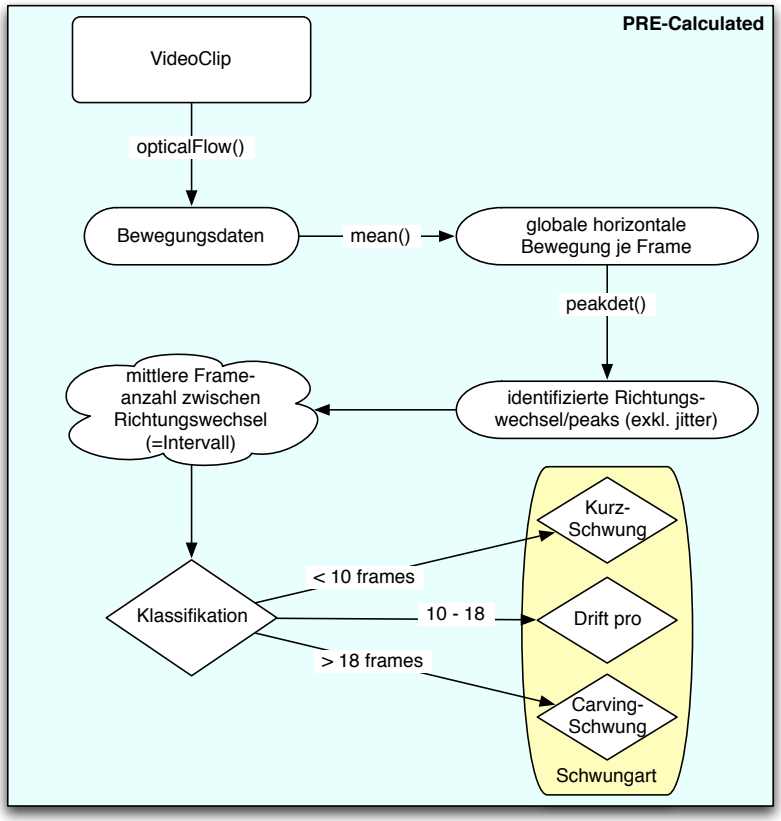

Abbildung 10: Modell des Gesamtprozesses Klassifikation nach Schwungart

der gesamten Datenreihe (globale Bewegungsvektoren) definiert.

Leicht ermitteln lässt sich dann das gesuchte (durchschnittliche) Schwungintervall, das durch die durchschnittliche Anzahl der Frames (= Dauer) zwischen zwei Peaks definiert ist, bei Kurz-Schwüngen eher klein und bei Carving-Schwüngen größer ist (siehe Vergleich in Abbildung 11).
Nochmals zur Verdeutlichung: Was wir hier sehen ist die Darstellung der Grundbewegung auf der horizontalen Achse (links-rechts Bewegung) je Frame. Gut zu erkennen ist, dass die Richtung sehr oft wechselt (inkl. Wackeln/-

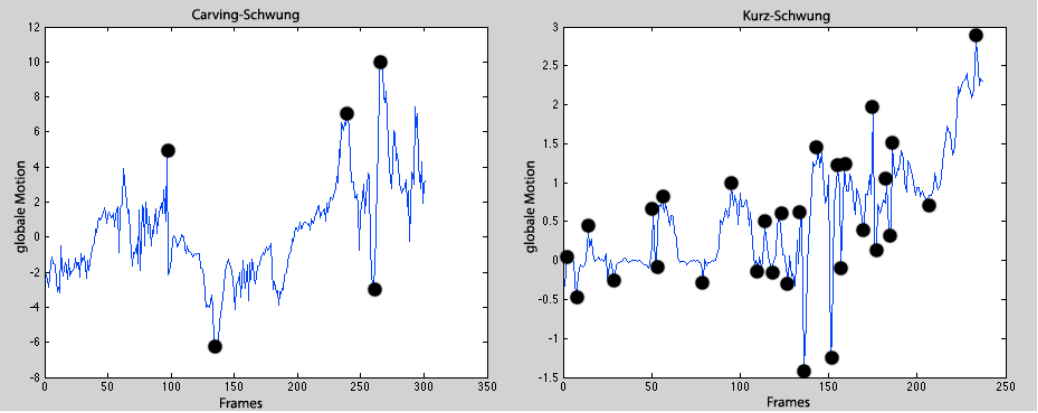

Abbildung 11: Vergleich des unterschiedlichen Schwungintervalls bei einem Carving- bzw. Kurz-Schwung

Zittern durch den Kameramann); die schwarz markierten Wechsel stellen jedoch schon die bereinigten „echten“ Richtungswechsel des Snowboarders dar und die Anzahl an Frames dazwischen kann als Maß für die Dauer eines Schwungs interpretiert werden.

3.2 Teilaufgabe: Klassifikation nach Personen

Die zweite Herausforderung, die im Rahmen dieser Arbeit behandelt wurde, ist die Klassifikation der Videoclips nach Personen.

3.2.1 theoret. Überlegungen

Das Hauptunterscheidungsmerkmal von Personen in solchen Szenen (auf Schneepiste) sind die Farben der Kleidung. Weder Gesicht noch andere Unterscheidungsmerkmale sind bei diesen Szenen ausschlaggebend oder weiterführend (da kaum sichtbar).
Das bedeutet, dass der Schlüssel zur korrekten Sammlung von Clips ein und der selben Person, eine Distanzberechnung zwischen den extrahierten Farbinformationen ist.
Wie schon in Kapitel 2.2.3 angedeutet, würde die Anwendung eines Farbhistogramms oder Clustering-Verfahrens (wie DominantColor) auf das Originalbild (inkl. Schnee) kaum nennenswerte Unterschiede zwischen den einzelnen Bildern aufzeigen (da Personen meist klein im Bild und Schnee vorherrschend). Daher muss zunächst der interessante Teil - der Snowboarder - vom Hintergrund getrennt und in weiterer Folge der Farbanalyse übergeben werden.

3.2.2 Implementierung

Kernkomponente dieser Lösung stellt der - zuvor schon in Kapitel 2.1.1 vorgestellte - Dominant Color Descriptor (DCD), der im Zuge des MPEG-7 Standards entwickelt wurde, dar. Nochmals zusammengefasst beschreibt dieses Verfahren die dominierenden Farben in einem Bild und ermöglicht weiters eine Distanzberechnung.

Wie im Modell (Abbildung 12) zu sehen, kommt der DCD zweimal zur Anwendung. Beim ersten Mal ist seine Aufgabe die *zwei* dominierenden Far-

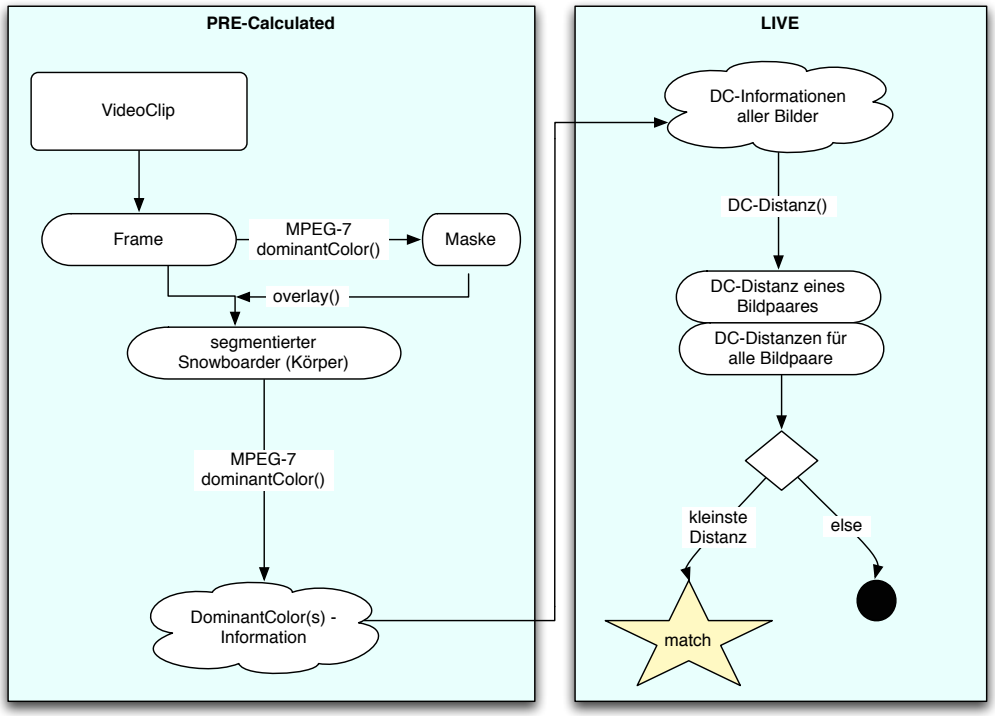

Abbildung 12: Modell des Gesamtprozesses Klassifikation nach Personen

ben (Schnee und Snowboarder) zu finden und eine binäre Bildmaske für jede zurückzugeben. Da die Anzahl der Pixel, die zum Schnee gehören meist überwiegt, wird mit *der* Maske fortgefahren, die den größten Teil (nämlich den Schnee) „ausblendet“ (siehe Abbildung 13).

Abbildung 13: Maske: eines der beiden binären Bildern (ermittelt durch DCD)

Die selektierte Maske wird über das Originalbild gelegt und stellt nun den Snowboarder frei, was in 75% aller Fälle zu guten Ergebnissen führt (siehe Beispiele in Abbildung 14). Die anderen Bilder weisen eine zu starke Komplexität für dieses Verfahren auf, da mehr als zwei dominierende Farben existieren (Bäume, Himmel, andere Personen,...), und wurden aufgrund ihrer deutlichen Minderheit vom Testset ausgeschlossen.

Das nun vorliegende Bild eines freigestellten Snowboarders fließt nun zum zweiten Mal in die Berechnungen des DCD ein und zwar wieder mit der Vorgabe, dass die *zwei* dominierenden Farben gefunden werden sollen. Diese Vorgabe könnte auch anders aussehen, doch wird davon ausgegangen, dass die Snowboarder unterschiedliche Farben am Ober- und Unterkörper tragen. Hier wird der Vorteil gegenüber einer bloßen Farb-Mittelwertsbestimmung deutlich, welche nicht nach zwei Outfit-Farben unterscheiden würde.

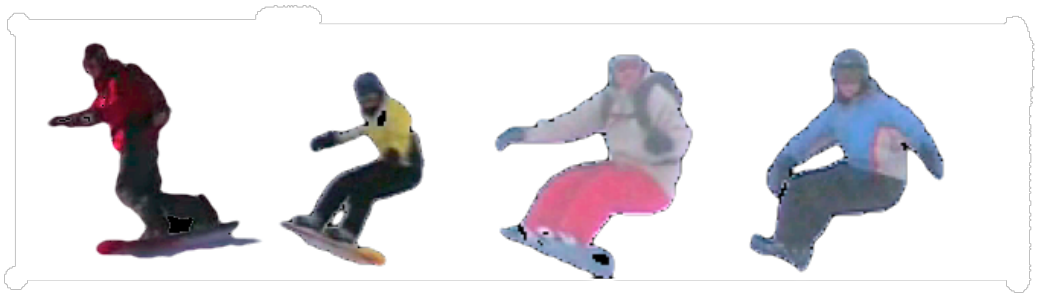

Abbildung 14: beispielhafte Ergebnisse der Segmentierung

Nun interessiert jedoch nicht eine Maske, sondern wirklich die ermittelten Daten über die beiden Dominant Colors. Der DCD stellt dazu unterschiedliche Auswertungen zur Verfügung. Zum einen die DC als RGB-Werte, aber auch die prozentuelle Verteilung und ein Maß für die Varianz der beiden Farben.
Dieser DCD wird für jedes Bild vorab kalkuliert und in einer geeigneten Datenstruktur archiviert.
Zur Laufzeit (des GUI) wird dann zu jedem Bild der „nearest neighbour“ - also das Bild mit der niedrigsten Distanz (aufgrund der Dominant Color Distance; ebenfalls durch MPEG-7 definiert) - gesucht. Somit können Bilder ein und der selben Person als solche gekennzeichnet und dem User im GUI dementsprechend gesammelt dargestellt werden.

4 Experimente

Um darzulegen, wie gut sich die Wahl und Kombination der Methoden herausgestellt hat, wurden die Algorithmen an einem Testset aus Videoclips ausgeführt und die Ergebnisse zusammengetragen.

4.1 Testset - Beschreibung des Videomaterials

Das Testset bestand aus 20 Videosequenzen mit einer durchschnittlichen Spiellänge von zehn Sekunden. Das ursprüngliche Video wurde mit einer DigiCam aufgezeichnet, auf einen Computer übertragen und in einzelne Abfahrtsszenen zerstückelt. Wie schon zuvor angesprochen, wurden die Clips in weiterer Folge mit einem Bildstabilisierungs-Programm von störenden Erschütterungen befreit; ansonsten aber nicht manipuliert.

4.2 Ergebnisse

Folgend sollen nun die Ergebnisse des Prototyps getrennt nach den beiden Hauptaufgaben dargelegt und kommentiert werden.

4.2.1 Ergebnis: Klassifikation nach Schwungarten

Das Resultat der Berechnung des Schwungintervalls (zur Bestimmung der Schwungart) ist durchwegs zufriedenstellend: 85% aller vorliegenden Test-Clips wurden nicht nur richtig klassifiziert, der ausschlaggebende Wert (des durschnittlichen Schwungintervalls) weist bei 65% sogar eindeutig auf die entsprechende Schwungart hin (siehe grün markierte Werte in Abbildung 15).
Die fehlerhaften Zuordnungen lässen sich auf zwei Hauptfehlerquellen zurückführen:

- einerseits auf die - zuvor in Kapitel 3.1.2 erwähnte - jitter-Problematik. Carving-Clips werden falscher Weise als Kurz-Schwünge identifiziert, da einfach zu viele/starke Verwackelungen im Clip vorkommen und

Filename	Intervall
anna_carv	53
sabrina_carv	50
susi_carv	47
dominik_carv	27
sadi_carv	26
joachim_carv	25
sadi_kurz	24
sadi_driftpro	21
dominik_kurz2	20
pia_carv	19
andi_carv	18
pia_driftpro	10
nicola_carv	9
dominik_kurz	9
andi_kurz	8
susi_kurz	8
anna_kurz	7
sabrina_kurz	4
joachim_kurz	3
nicola_kurz	3

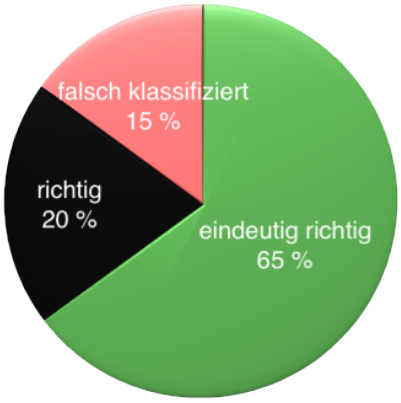

Abbildung 15: deutlich zu erkennende Unterschiede zwischen carv und kurz Clips. Grün markierte Clips wurden eindeutig richtig, schwarze korrekt und rote falsch klassifiziert.

somit die Auswertung des Bewegungsintervalls hinfällig ist (vgl. Intervallwert des Clips `nicola_carv` in Abbildung 15),

- andererseits auf einen aufnahmetechnischen Grund, welcher vor allem die fehlerhafte Klassifizierung von Kurz-Schwüngen als Carving-Schwünge erklärt (vgl. `dominik_kurz2` und `sadi_kurz` in Abb.15): Die Snowboarder wurden manchmal komplett ohne Zoom im Weitwinkel aufgenommen. Die dabei entstandenen minimalen Bewegungen/Richtungswechsel wirken sich für das System zu geringfügig aus, um als dynamische, schnell-rhythmische Bewegung interpretiert zu werden. Diesem Problem könnte man jedoch mit einem kleineren Ausschnitt, bei dem sich die Analyse dann auf das Wesentliche konzentriert, entgegenwirken.

Ergänzend noch ein paar Worte zu den beiden Clips `sadi_driftpro` und `pia_driftpro`: Die Ausführung des „Driftschwung Pro“ ist nicht an einen

bestimmten Rythmus oder Geschwindigkeit gebunden. Es kommt dabei mehr auf Körperhaltung und Einsatz der Snowboardkante an. Somit kann/darf es sein, dass driftpro Clips nicht eindeutig zugeordnet werden.

4.2.2 Ergebnis: Klassifikation nach Personen

Im Gegensatz zu den erfolgreichen Ergebnissen der Klassifikation nach Schwungarten, sind die der Personendifferenzierung nur bedingt zufriedenstellend. Prinzipiell wird vom Algorithmus das getan, was erwünscht ist, nämlich eine Klassifikation aufgrund Farbdifferenzen. Dies funktioniert bei einem sehr kleinen Testset auch sehr gut.
Bei wachsender Anzahl an Bildern/Personen, die in die Berechnungen bzw. die Suche nach dem nearest neighbour eingebunden werden (vgl. Abb. 17), steigt jedoch die Wahrscheinlichkeit, dass aufgrund ähnlicher (Kleidungs-) Farben bzw. unterschiedlicher Lichtverhältnisse (schattige/sonnige Pistenabschnitte) Bilder falsch gepaart werden. So kann es vorkommen, dass ein und die selbe Person (und gleicher Kleidung) auf einem Clip völlig andere Farben zurückwirft, als auf einem anderen (vgl. Unterschied in Abb. 16); dafür aber eventuell hohe Ähnlichkeit mit einer anderen Person besitzt.

Abbildung 16: Farben wirken in unterschiedlichen Lichtverhältnissen anders. Daher ist ein Vergleich oft schwierig und birgt die Gefahr, dass die Dominant Colors anderer Personen wesentlich besser übereinstimmen, was in einer falschen Paarung resultiert.

Auch wenn das menschliche Auge - beim Vergleich der beiden Beispiel-Fotos - den Unterschied instinktiv „wegrechnet“, für den Algorithmus der Dominant

Color Distance sind die Farben einfach zu unterschiedlich.

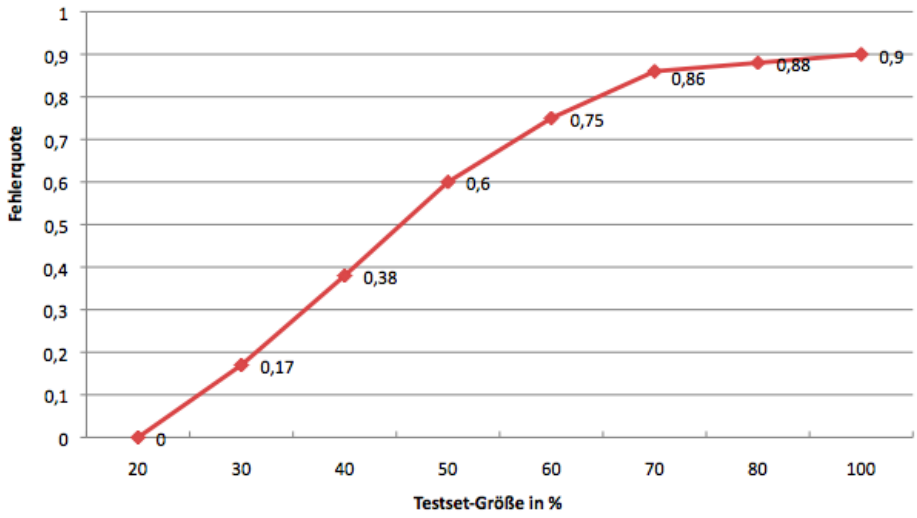

Abbildung 17: zeigt das Verhältnis zwischen Testset-Größe (Anzahl der Clips, die in die Berechnung miteinbezogen wurden) und der dabei entstandenen Fehlerquote.

Es könnte nun bei weiteren Experimenten und Erweiterung des Algorithmus versucht werden, alle Clips in einem ersten Schritt in ihrer Farbe zu normalisieren. Dies könnte durch einen Weißabgleich-Filter erreicht werden, der die Farbwerte jedes einzelnen Clips um einen individuellen Faktor `d` korrigiert. `d` ergibt sich dabei aus der Distanz zwischen reinem Weiß `rgb(0,0,0)` und dem ermittelten Schneeweiß des einzelnen Clips. Dadurch würden die Farben aller Clips auf ein gleiches Level normalisiert und das Ergebnis aussagekräftiger sein.
Abschließend muss resümiert werden, dass die Vorgehensweise noch nicht die gewünschte Stabilität mitbringt, um ein aussagekräftiges Ergebnis bei der Analyse eines größeren Testsets zu erhalten. Die Ergebnisse bei einer geringen Anzahl an Testclips lassen aber durchaus die Aussage zu, dass die Methodik grundsätzlich funktioniert und weiter verfolgt werden könnte.

5 Schlussfolgerungen und Fazit

Die gesetzte Aufgabenstellung einer automatisierten Kategorisierung von Snowboardclips war von Anfang an sehr speziell ausgerichtet, die nur in ihren Ansätzen und theoretischen Grundlagen für andere Bereiche relevant ist.
Die theoretischen Überlegungen (aus fachlicher Sicht), beispielsweise die Logik der Schwungarten-Erkennung, waren großteils von Anfang an richtig und konnten mit Hilfe bereits bestehender Techniken erfolgreich umgesetzt werden.

Als wirklich große Herausforderung bzw. Einschränkung für ein mögliches Endprodukt jedoch, stellte sich die Tatsache heraus, dass der nun vorliegende Lösungsvorschlag nur unter gewissen Voraussetzungen befriedigende Ergebnisse liefert. So war es mir in der Konzeptionsphase noch nicht wirklich bewusst, dass Snowboardaufnahmen aus dem täglichen Schulungsalltag selten einer Studioaufnahme entsprechen, was zusätzlich Komplexität in die Anforderungen an ein Kategorisierungssystem bringt. Denn Lichtverhältnisse und die Komplexität des Bildaufbaus (zusätzliche Objekte wie Bäume, andere Personen,...) sind nicht zu unterschätzende Rahmenbedingungen, die einen entsprechend hohen Umsetzungsaufwand für ein Endprodukt zur Folge hätten.

Literatur

[1] R. M. S. Adrienne E. Hunt. Mechanics and control of the flat versus normal foot during the stance phase of walking. *Clinical biomechanics (Bristol, Avon)*, 19(4):391–397, 1975.

[2] A. Baraldi and F. Parmiggiani. An investigation of the textural characteristics associated with gray level cooccurrence matrix statistical parameters. *Geoscience and Remote Sensing, IEEE Transactions on*, 33(2):293–304, Mar 1995.

[3] D. B. Brown. Motion-based foreground segmentation, 2000.

[4] G. Cortelazzo. Trademark shapes description by string-matching techniques. *Pattern Recognition*, 27(8):1005–1018, August 1994.

[5] H. Eidenberger. A new perspective on visual information retrieval. In *SPIE Electronic Imaging Symposium*, page 2004, 1997.

[6] H. Eidenberger. Suchmodellbasiertes content-based image retrieval: hnlichkeitsdefinition, anwendung und automatisierung, 2000.

[7] H.-P. Frei, S. Meienberg, and P. Schäuble. The perils of interpreting recall and precision values. In *Proceedings of the GI/GMD-Workshop on Information Retrieval*, pages 1–10, London, UK, 1991. Springer-Verlag.

[8] G. Gupta and C. Chakrabarti. Architectures for hierarchical and other block matching algorithms. *IEEE Transactions on Volume 5, Issue*, 1995.

[9] B. Haskell, A. Puri, and A. Netravali. *Digital video: an introduction to MPEG-2*. Kluwer Academic Publishers, 1996.

[10] B. K. Horn and B. G. Schunck. Determining optical flow. Technical report, Cambridge, MA, USA, 1980.

[11] M.-K. Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, February 1962.

[12] A. K. Jain and A. Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 29:1233–1244, 1996.

[13] S. Khan and M. Shah. Object based segmentation of video using color, motion and spatial information. In *IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION*, volume 2. IEEE Computer Society; 1999, 2001.

[14] D. Le Gall. Mpeg: a video compression standard for multimedia applications. *Commun. ACM*, 34(4):46–58, 1991.

[15] R. Li, B. Zeng, and M. Liou. A new three-step search algorithm for block motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 4(4):438–442, Aug 1994.

[16] J. Lu and M. Liou. A simple and efficient search algorithm for block-matching motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 7(2):429–433, Apr 1997.

[17] B. D. Lucas and T. Kanade. 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)*, pages 674–679, April 1981. A more complete version is available as Proceedings DARPA Image Understanding Workshop, April 1981, pp.121-130. When you refer to this work, please refer to the IJCAI paper.

[18] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel. Performance measures for information extraction. In *In Proceedings of DARPA Broadcast News Workshop*, pages 249–252, 1999.

[19] J. M. Martnez. Mpeg-7 overview (standard). Technical report, International Organisation for Standardisation, 2004.

[20] M. Maruya, K. Nemoto, and Y. Takashima. Texture based 3d shape reconstruction from multiple stereo images. In *Pattern Recognition, 1992. Vol.I. Conference A: Computer Vision and Applications, Proceedings., 11th IAPR International Conference on*, pages 137–140, Aug-3 Sep 1992.

[21] B. M. Mehtre, M. S. Kankanhalli, and W. F. Lee. Shape measures for content based image retrieval: a comparison. *Inf. Process. Manage.*, 33(3):319–337, 1997.

[22] S. Negahdaripour. Revised interpretation of optical flow for dynamic scene analysis. *Computer Vision, International Symposium on*, 0:473, 1995.

[23] A. S. Ogale, 2006. `http://www.cs.umd.edu/users/ogale/download/code.html`.

[24] C. Palm. *Integrative Auswertung von Farbe und Textur*. Der Andere Verlag, D-52062 Aachen, 2003.

[25] T. Pavlidis. *Algorithms for Graphics and Imag*. W. H. Freeman & Co., New York, NY, USA, 1983.

[26] S. Peleg and H. R. T. Motion based segmentation. In *In International Conference on Pattern Recognition*, pages 109–113, 1990.

[27] L.-M. Po and W.-C. Ma. A novel four-step search algorithm for fast block motion estimation. *IEEE Trans. Circuits Syst. Video Technol*, 6:313–317, 1996.

[28] J. Radanitsch. *Spezielle Bewegungslehre und Methodik, Instruktor Snowboarden*. Bundesanstalt für Leibeserziehung Graz.

[29] P. U. Radanitsch Jan, Radanitsch Jörg. *Snowboarden leicht gemacht*. Österr. Alpenverein, Edition Berg & Steigen.

[30] T. R. Reed and J. M. H. du Buf. A review of recent texture segmentation and feature extraction techniques. *CVGIP: Image Underst.*, 57(3):359–372, 1993.

[31] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, 2000.

[32] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool for interactive content-based image retrieval, 1998.

[33] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[34] H. Shu, L. Luo, and J.-L. Coatrieux. Moment-based approaches in imaging. 1. basic features [a look at ...]. *Engineering in Medicine and Biology Magazine, IEEE*, 26(5):70–74, Sept.-Oct. 2007.

[35] T. Sikora. The mpeg-7 visual standard for content description-an overview. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6):696–702, 2001.

[36] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, 2000.

[37] M. J. Swain. *Color indexing*. PhD thesis, 1990. Supervisor-Ballard, Dana H.

[38] C. van Rijsbergen and P. D. Information retrieval, 1979.

[39] R. Yared, X. Defago, and M. Wiesmann. Collision prevention using group communication for asynchronous cooperative mobile robots. *Research report*, 2007:1–21, 20070222.

[40] D. Zhang and G. Lu. A comparative study on shape retrieval using fourier descriptors with different shape signatures. *Journal of Visual Communication and Image Representation*, (14 (1)):41–60, 2003.

A Anhang

A.1 Source Code

Auf den nun folgenden Seiten ist der Source Code der wichtigsten Funktionen des Prototyps einsehbar.

A.1.1 classification.m - Klassifikation zur Laufzeit

```
% function classification()
% computes classification of clips and saves results in
    classified[]
%
% OUTPUT: classified[]: dataStruct
%
% Author / Copyright: Dominik Lepizh, 2007.

function classified = classification()
clc
classified = [];

%########## first compute classification "turn type
    "##############

% local path to precalculated average-motion-data-files
d = dir ('/Users/dominiklepizh/Documents/Studium/Studium_Faecher
    /Mag. Arbeit/Work/matlab/medfiles/*.mat');
cd /Users/dominiklepizh/Documents/Studium/Studium_Faecher/Mag.
    Arbeit/Work/matlab/medfiles

for k = 1:length(d)

      filename = strrep(d(k,1).name, '_steady.mat', '');
      % load precalculated average-motion-data for each clip
      load(d(k,1).name);

      % detect peaks in average motion
      [maxtab, mintab]=peakdet(med, var(med));

      % count (average) number of frames between two peaks
      [med_frames] = get_frameinfo(maxtab,mintab);

      % do classification based on (average) number of frames
          between two peaks
```

```
    if (med_frames < 10)
       turnclass = 2;                    % it's a short turn
    elseif (med_frames >=10 && med_frames <19)
       turnclass = 3;                    % it's a drift pro
    elseif (med_frames >=19)
       turnclass = 1;                    % it's a carving turn
    end

    % build datastruct with results
    classified(k).filename = filename;
    classified(k).turnclass = turnclass;
end


%########## second compute classification "persons
    "##############

    % defining a testset for color distance measurement
    % testset: 1 = involve in calculations
    testset(1) = 1;      % andi_carv      1
    testset(2) = 1;      % andi_kurz      1
    testset(3) = 1;      % anna_carv      1
    testset(4) = 1;      % anna_kurz      1
    testset(5) = 1;      % dominik_carv   1
    testset(6) = 1;      % dominik_kurz   0
    testset(7) = 1;      % dominik_kurz2  1
    testset(8) = 1;      % joachim_carv   1
    testset(9) = 1;      % joachim kurz   1
    testset(10) = 1;     % nicola_carv    0
    testset(11) = 1;     % nicola_kurz    0
    testset(12) = 1;     % pia_carv       1
    testset(13) = 1;     % pia_driftpro   1
    testset(14) = 1;     % sabrina_carv   0
    testset(15) = 1;     % sabrina_kurz   0
    testset(16) = 1;     % sadi_carv      1
    testset(17) = 1;     % sadi_driftpro  1
    testset(18) = 1;     % sadi_kurz      0
    testset(19) = 1;     % susi_carv      0
    testset(20) = 1;     % susi_kurz      0



% local path to precalculated dominantColor-featureVectors
d = dir ('/Users/dominiklepizh/Documents/Studium/Studium_Faecher
    /Mag. Arbeit/Work/matlab/featureVec/*.mat');
```

```
cd /Users/dominiklepizh/Documents/Studium/Studium_Faecher/Mag.
    Arbeit/Work/matlab/featureVec

for k = 1:length(d) %#ok<USENS>

    if (testset(k) ~= 0)
        % load precalculated dominantColor-featureVector for
            each clip
        filename = d(k,1).name;
        descr1 = load(filename);
        descr1 = descr1.descr;

        temp_dist = 1;
        temp_index = 0;
        pid = 1;

        % search nearest neighbours based on dominantColor
            distance
        for l = 1:length(d)
            filename2 = d(l,1).name;
            descr2 = load(filename2);
            descr2 = descr2.descr;

            if (k~=l && testset(l) ~= 0)
                distance = DCD(descr1,descr2);
                classified(k).persons(pid,1) = l;
                classified(k).persons(pid,2) = distance;
                pid = pid + 1;

                if (distance < temp_dist)
                    temp_dist = distance;
                    temp_index = l;
                end
            end
        end

        % build datastruct with results
        classified(k).persons = sortrows(classified(k).persons
            ,2);
        classified(k).person = temp_index;
    else
        classified(k).person = [];
    end
end
```

A.1.2 get_frameinfo.m - Ermittlung des Schwungintervalls

```
function [med_frames] = get_frameinfo(max,min)

% Vergleich der Peaks

first_max_index = max(1,1);
first_min_index = min(1,1);
frames = [];
j = 0;

% bestimme welches array den ersten Frame beinhaltet
if (first_max_index < first_min_index)
    first = max;
    second = min;
else first = min;
     second = max;
end

% bestimme das gr ere Array und speichere size
if (size(min,1) < size(max,1))
    maxsize = size(max,1);
else maxsize = size(min,1);
end

frames(1) = abs(0 - first(1,1));


% berechne Anzahl der Frames zwischen zwei peaks
for i=1:maxsize-1
    j=j+2;
    frames(j) = abs(first(i,1) - second(i,1));
    frames(j+1) = abs(first(i+1,1) - second(i,1));

end


med_frames = mean(frames);
```

A.1.3 DCD.m - DominantColor - Distanz

```
% function DCD()
% perform DominantColor-Distance Calculation
%
% INPUTS: - descr1: [dataStruct], dominantColor-Information from
    image1
%         - descr2: [dataStruct], dominantColor-Information from
    image2
%
% OUTPUTS: - distance: distance between image1 and image2 based
   on dominantColor-Distance
%
% Author / Copyright: Dominik Lepizh, 2007.
% based on MPEG-7 DominantColorDescriptor


%########## ALGORITHM ###################################
function distance = DCD(descr1, descr2)

w1 = 0.3;       % weight term1
w2 = 0.7;       % weight term2

% spatial coherency diff.
sc_diff = abs(descr1(1).spatial - descr2(1).spatial);

% dominant colors diff.
dc_diff = dcdiff(descr1,descr2);

% Distance Function
distance = (w1 * sc_diff * dc_diff) + (w2 * dc_diff);


%######### DC_DIFF
   ######################################################
function dc_diff = dcdiff(descr1,descr2)

%          |    TERM 1    | + |    TERM 2    | - |       TERM 3       |
dc_diff = term12(descr1) + term12(descr2) - term3(descr1,descr2)
    ;


%######### TERM 1 and 2
   ######################################################
function result = term12(descr)
```

```
result = 0;

for i=2:length(descr)
    result = result + (descr(i).percent * descr(i).percent);
end

%######### TERM3
    ##################################################
function result = term3(descr1,descr2)

Td = 20;
a = 1;
dmax = a*Td;          % Threshold
sum = 0;

for i = 2:length(descr1)
    EuclideanVec = (descr1(i).rgb - descr2(i).rgb);
    eucldist = sqrt(EuclideanVec * EuclideanVec');

    if (eucldist <= Td)
        akl = 1 - (eucldist / dmax);
    else
        akl = 0;
    end

    midsum = 2 * akl * descr1(i).percent * descr2(i).percent;
    sum = sum + midsum;
end
result = sum;
```

A.1.4 getsegmented.m - Extraktion des Snowboarders

```
% function getsegmented()
% segments snowboarder from background
%
% INPUT: image/frame
% OUTPUT: image (only pixels which form the person in the image;
    not the background)
%
% Author / Copyright: Dominik Lepizh, 2007.
function im = getsegmented(image)

param.numOfDominantColors=2;
param.visualize = false;
param.seed=0;

% get segmentation masks (performed by calculating
   dominantcolors)
[garbage,masks] = feature_image_MPEG7_DC(image, param);

numwhites = 999999999;
for i = 1:length(masks)
    [r,c,v] = find(masks{1,i});
    if (length(v) < numwhites)     % elicit mask with most
        substractions
        numwhites = length(v);
        index = i;
    end
end

mask = masks{1,index};
for y=1:size(mask,1)
    for x=1:size(mask,2)
        if (mask(y,x) == 0)        % invert mask
            mask(y,x) = 1;
        else mask(y,x) = 0;
        end
    end
end

color = [0 0 0];                   % cropped areas filled with
   color[]
im = imoverlay(image, im2bw(mask), color); % combine frameimage
   with mask
im = imcrop(im,[5 5 342 262]);     % crop border (5px)
```

```
im = withoutblackpixels(im);          % black pixels will be ignored
```