Freie Universität Berlin

Fachbereich Mathematik und Informatik
Institut für Informatik
Softwareprojekt Mobilkommunikation

Vordergrundextraktion auf Basis von Hidden Markov Modellen

Marcin Nawrocki
Patrick Mertes
Hinnerk van Bruinehsen

Betreuer: Alexandra Danilkina

28. Februar 2014

Inhaltsverzeichnis

1 Einleitung

Dieses Kapitel gibt eine Einführung in das Projekt SAFEST und erläutert die vorliegende Problemstellung, welche im Rahmen des Softwareprojektes Mobilkommunikation im Wintersemester 13/14 bearbeitet wurde.

1.1 Kontext und Motivation

SAFEST ist ein deutsch-französisches Projekt mit der Zielsetzung, die Sicherheit an öffentlichen Plätzen und kritischen Infrastrukturen durch Realisierung eines Sensornetzwerkes mit Infrarotkameras zu erhöhen. Um dieses Ziel zu erreichen, überwacht das Sensornetzwerk ein vorgegebenes Areal und verarbeitet die gewonnenen Mittelinfrarotbilder mit Hilfe von Algorithmen in Hinblick auf verschiedene Parameter. Der eigentliche Gewinn an Sicherheit soll dadurch erfolgen, dass automatisiert auf Grundlage der ermittelten Parameter die Dichte der sich in dem Areal aufhaltenden Personen ermittelt wird, woraus wiederum ein Rückschluss auf die Wahrscheinlichkeit des Auftretens einer Massenpanik gezogen werden soll.
Der erste wichtige Schritt bei dieser Analyse ist die automatisiere Erkennung und Zählung von sich im Bildbereich befindlichen Personen. Um diesen Schritt zu ermöglichen, muss zunächst der Hintergrund vom Vordergrund des Bildes getrennt werden. Die verwendete Kamera stellt wärmere Bereiche heller und kältere Bereiche dunkler dar. Daraus folgt, dass Personen erwartungsgemäß eher weiß und Hintergrund erwartungsgemäß eher schwarz dargestellt werden. Allerdings kalibiert sich die Kamera regelmäßig neu, um den gesamten Graustufenverlauf zur darstellung der Wärme zu nutzen. Daraus folgt, dass mitunter auch sehr kalte Bereiche sehr hell dargestellt werden, wenn diese die wärmsten von der Kamera erfassten Temperaturen besitzen. Aus dieser Eigenschaft der Kamera folgt somit auch, dass ein naiver Ansatz wie eine feste Zuordnung von Temperatur und Graustufenwert nicht möglich ist. Ein weiteres Hindernis bei der einfachen Zuordnung von Temperaturwerten zu Vorder- oder Hintergrund ist die Tatsache, dass Kleidung (besonders zum Beispiel dickere Jacken) die Temperatur sehr gut nach außen isolieren und damit Teile von Personen sehr dunkel auf dem Kamerabild erscheinen lassen.
Die Trennung von Vorder- und Hintergrund kann daher nicht allein aufgrund der dargestellten Temperatur erfolgen, sondern muss als zweiten Faktor die Bewegung mit einbeziehen. Personen sind also in der Darstellung der Regel nach helle, bewegte Objekte.
Nach erfolgter Entfernung des Hintergrunds folgt als nächster Schritt in der Analyse die automatisierte Bestimmung der auf dem Bild sich befindlichen Personen. Wird diese Analyse erfolgreich ausgeführt, so muss nur noch dieser Wert an eine Kontrolleinheit übertragen werden, nicht aber das eigentliche Bild, was im Hinblick auf auf den Datenschutz sehr wichtig ist.
Eigentlich für Objekterkennung bewährte Algorithmen wie Histogramm orientierte Gradienten (HOG)[1] und die Mischung gausscher Verteilungsdichten (MOG)[2], können sich auf die Eigenschaften von Vorder- und Hintergrund, insbesondere die durch die Kamerakalibrierung entstehende Dynamik, nicht so einstellen, dass sie befriedigende Ergebnisse liefern.

Daher soll ein neuer Algorithmus entwickelt und evaluiert werden, der über die Fähigkeit verfügt, sich der Dynamik anzupassen, um das Problem zu lösen.

1.2 Aufgabenbeschreibung

Das Ziel unseres Softwareprojektes ist es, ein Verfahren zu entwickeln, welches das Bild einer Infrarotkamera in Vorder- und Hintergrund trennt. Die genauen Anforderungen für diesen Prozess werden in Kapitel 2. 1 beschrieben. Hierbei soll die allgemeine Verwendbarkeit von Hidden-Markov-Modellen(HMM)[3] und der Diskreten Kosinustransformation(DCT)[4] gezeigt werden.. Die theoretischen Grundlagen werden hierzu im Kapitel 2 ff. erläutert. Schlussendlich soll eine vergleichende Evaluation durchgeführt werden, in dem das entwickelte Verfahren gegen ein bereits vorliegendes Histrogramm-basiertes Verfahren antreten muss(vgl. Kapitel 4).
Da bereits ein Algorithmus zum Zählen von Personen vorliegt, muss dieser nicht innerhalb dieses Projektes entworfen beziehungsweise implementiert werden.

2 Entwurf eines Vordergrundextraktors

In diesem Kapitel wird der nötige, theoretische Hintergrund herausgearbeitet und unser Entwurf gemäß den Anforderungen vorgestellt. Es wird das Konzept der HMM und der DCT erläutert und deren Verknüpfung eingeführt.

2.1 Anforderungsanalyse

Die funktionalen Anforderungen an einen Vordergrundextraktor sind bereits in der Aufgabenbeschreibung (Kapitel 1.2) ausgeführt: Es soll demonstriert werden, dass HMM zur Trennung von Vordergrund und Hintergrund bei stark heterogenen Hintergründen geeignet sind, wie in Abbildung 1 dargestellt.

Abbildung 1: Kamerabild mit heterogenen Hintergund, Vordergrund ist markiert.

Die nichtfunktionalen Anforderungen umfassen vor allem eine einfache Benutzbarkeit durch Automatisierung. Der Ressourcenbedarf soll möglichst gering sein - ein Video wird für gewöhnlich mit 25 Frames pro Sekunde aufgezeichnet, im Idealfall ist eine optimale Analyse jedes einzelnen Frames in Echtzeit möglich. Allerdings ist zu berücksichtigen, dass eine möglichst hohe Korrektheit wichtiger ist, als eine Analyse aller Frames, da relevante Informationen wie Eintritt bzw. Austritt von Personen aufgrund der relativ langsamen Bewegung der Menschen auch von den nachfolgenden Frames erfasst werden. Falls nötig können demnach Frames von der Analyse ausgeschlossen werden. Die Analyse sollte jedoch unabhängig von den Parametern des verwendeten Videomaterials wie beispielsweise unterschiedliche Videoformate oder Auflösungen statt finden.

2.2 Hidden-Markov-Modelle

Die Markov Modelle stammen aus der Wahrscheinlichkeitstheorie und entsprechen einem stochastischem Zustandsautomaten, bei dem die Zustandswechsel gemäß einer Wahrscheinlichkeit statt finden und nicht abhängig von der Vergangenheit sind, sondern nur von dem aktuellen Zustand[3]. Bei den HMM können jene Zustände nicht beobachtet werden, sondern nur die Beobachtungen/Ausgaben, welche während dieses Zustandes

auftreten, sie werden Emissionen genannt. Die Zustandsübergangswahrscheinlichkeiten sind in den HMM somit nicht die einzigen Parameter, die Emissionswahrscheinlichkeiten bilden die zweiten Parameter.
Formal definiert ist ein HMM ein 5er-Tupel $\lambda = (S, V, A, B, \Pi)$ mit:

- $S = s_1, ..., s_n$ sei die Menge aller Zustände
- $V = v_1, ..., v_m$ das Alphabet der möglichen Emissionen
- $A \in \mathbb{R}^{n \times n}$ sei eine Übergangsmatrix zwischen den Zuständen, a_{ij} entspricht der Wahrscheinlichkeit des Übergangs von Zustand s_i in Zustand s_j
- $B \in \mathbb{R}^{n \times m}$ sei eine Beobachtungsmatrix, wobei $b_i(v_j)$ die Wahrscheinlihkeit angibt, im Zustand s_i die Beobachtung (Emission) v_j zu sehen
- $\Pi \in R^n$ die Initialverteilung, Π_i sei die Wahrscheinlichkeit, dass s_i der Startzustand ist

Ein HMM heißt zeitinvariant, wenn sich die Übergangs- und Emissionswahrscheinlichkeiten nicht mit der Zeit ändern.
HMM werden zunehmend in der Literatur zur Sprach-, Schrift und Mustererkennung [5][6] verwendet, da sie mit den probabilistischen Übergängen die Prozesse der echten Welt besser widerspiegeln als deterministische Definitionen, zu dem können die Parameter durch Lernalgorithmen automatisiert bestimmt werden, siehe hierzu Kapitel 2.3.

Zum besseren Verständnis der vorangehenden Definition visualisiert Abbildung 2 ein exemplarisches HMM. Es liegt ein HMM mit 3 Zuständen und 4 Emissionen vor, wobei der Übergang aus jedem Zustand zu jedem Zustand und zusätzlich in jedem Zustand jede Emission möglich ist. In diesem Beispiel liegt eine Gleichverteilung vor, Übergänge zwischen Zuständen haben stets die Wahrscheinlichkeit 1/3, die Emissionen haben stets eine Wahrscheinlichkeit von 1/4. Beschriftung der Kanten aus Übersichtsgründen nicht dargestellt.

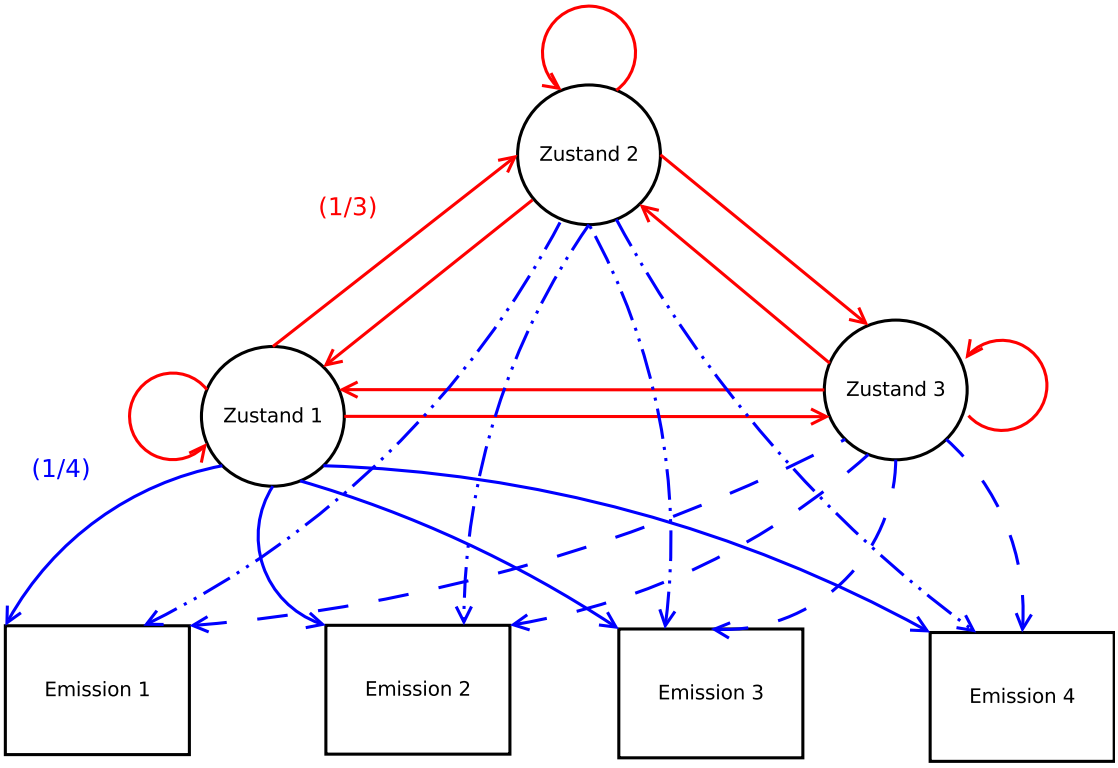

Abbildung 2: Beispiel für ein HMM, welches eine Gleichtverteilung aufweist. Eine Gleichverteilung wird oft initial angenommen und dann ein Lernprozess ausgeführt.

2.3 Gängige Problemstellungen der HMM

Für die HMM wurden bereits viele Algorithmen entwickelt, welche Standardprobleme lösen. Die zwei häufigsten Problemstellungen bei HMMs sind das Lernproblem und das Evaluations-/Decodingproblem.

1. Das Lernproblem bezieht sich auf die Problematik ein HMM anhand von Emissionssequenzen zu trainieren. Gelöst wird dieses Problem durch den Baum-Welch Algorithmus, welcher dazu verwendet wird, unbekannte Parameter, genauer die Übergangs- und Emiossionswahrscheinlichkeiten eines HMM, zu bestimmen. Hierbei handelt es sich um einen erwartungsmaximierenden Algorithmus, welcher anhand von übergebenen Trainingssequenzen die Maximum-Likelihood-Schätzwerte berechnet. Die initialen Werte eines HMMs müssen geschätzt werden, hier wird für gewöhnlich eine Gleichverteilung angenommen, dass heißt das jeder Übergang und jede Emission in einem Zustand gleich wahrscheinlich sind, vergleiche Abbildung 2.

2. Das Evaluations-/Decodingproblem bezieht sich auf die Problemaik aus den ver-

schieden Wahrscheinlichkeiten Aussagen über die Folgezustände oder aber über die Wahrscheinlichkeit bestimmter Zustandsketten zu treffen. Es wird durch den Forward-Algorithmus beziehungsweise den eng verwandten Viterbi-Algorithmus gelöst. Gegeben sei eine Chronik an k-letzten Emissionen, was ist die Wahrscheinlichkeit für eine bestimmte Emission? Und ferner, was ist die wahrscheinlichste Emission? Der Viterbi-Algorithmus berechnet zur Beantwortung dieser Frage die wahrscheinlichste Zustandssequenz, also eine Sequenz die die Wahrscheinlichkeit der übergebenen Emissionssequenz maximiert. Der Forward-Algorithmus ist ein Algorithmus aus der dynamischen Programmierung und optimiert im Gegensatz zum Viterbi nicht rückwirkend die gesamte Zustandssequenz neu, sondern berechnet den aktuell wahrscheinlichsten Zustand auf Basis der zuvor berechneten Zustände und hängt diesen an. Somit ist der Forward Algorithmus zur Laufzeit grundsätzlich schneller, jedoch nicht so genau wie der Viterbi-Algorithmus.

Anhand des gewonnen letzten Zustandes kann nun bestimmt werden, welche Emission als nächstes vorhergesagt wird. Hierbei wird aus den möglichen Emissionen des letzten Zustandes deterministisch die wahrscheinlichste Emission gewählt oder aber gemäß der Wahrscheinlichkeiten gewichtetet gewürfelt werden. Die zweite Variante aufgrund ihrer probabilistischen Natur ist stärker an den HMM orientiert und sollte daher vorgezogen werden.

2.4 Diskrete Kosinustransformation

Die Diskrete Kosinustransformation (DCT)[4] ist ein Verfahren, welches zur verlustbehafteten Kompression von Daten verwendet wird, wobei die bekannteste Anwendung das Dateiformat JPEG ist. Ähnlich zu der diskreten Fourier Transformation wird eine Information mittels Frequenzen repräsentiert, wobei wie der Name bereits nahe legt nur Kosiunusfunktionen verwendet werden.

Es existieren mehrere Varianten der DCT. Wird jedoch die DCT auf Bildinformationen angewandt, speziell bei der JPEG-Kompression, so wird die zwei-dimensionale Typ II DCT verwendet, diese ist demnach die gängigste Variante. Die DCT erhält eine $n \times n$ Matrix und gibt eine Matrix der selben Größe zurück, wobei im JPEG-Standard 8×8 Pixel Matrizen betrachtet werden. Hierbei wird das Element $[1, 1]$ als DC-Wert bezeichnet und bildet den Durchschnittswert der Farben der betrachteten Matrix; die restlichen 63 Werte der Matrix sind ein Offset zu dem DC-Wert und kodieren somit den Unterschied zum DC-Wert innerhalb der Matrix. Diese Werte werden als AC-Werte bezeichnet.
Formal definiert ist die DCT eine lineare, invertierbare Funktion $f : \mathbb{R}^n \to \mathbb{R}^n$, welche N reellwertige Werte aus $x[n]$ in N reellwertige Werte nach $X[n]$ überführt:

$$X_k = \sum_{n=0 \to N-1} x_n \cos[\tfrac{\Pi}{N} + (n + \tfrac{1}{2})k] \text{ mit } k = 0, ..., N-1$$

Abbildung 3 zeigt das Ergebnis nach der Anwendung der DCT mit Einfärben der gesamten Blöcke in die jeweils berechneten DC-Werte.

Abbildung 3: Links: Originalbild, rechts: Bild nach Anwendung der DCT blockweise nach DC-Wert eingefärbt

2.5 Modellierung eines Eingabealphabets

In unserem Projekt sollen HMM benutzt werden, um eine Entscheidung zu treffen, ob ein Bildbereich zum Vorder- oder zum Hintergrund gehört. Nach dem das generelle Konzept der HMM in Kapitel 2.2 vorgestellt wurde, fehlt jedoch noch die Definition, wie eine Emission aussieht. Das Alphabet der Emissionen wird als Eingabealphabet bezeichnet, da mit dessen Hilfe das System beschrieben werden kann. Hierbei ist zu beachten, dass das Eingabealphabet endlich und diskret sein muss. Zusätzlich sollte das Eingabealphabet aus möglichst wenigen, aussagekräftigen Symbolen bestehen. Die Definition eines Eingabealphabets, also die Zuordnung von Daten zu bestimmten Emissionen, gehört daher in unserem Fall zu der Kernleistung bei der Verwendung der HMM.
Die im Kapitel 2.3 vorgestellte DCT wird von uns verwendet, um ein Eingabealphabet zu erzeugen, wobei wir die DCT auf fest definierte Bereiche anwenden. Wir übernehmen die aus der Bildkompression üblichen 8 x 8 Pixelblöcke zur Einteilung des vorliegenden Bildes, da wir dieses als optimalen Kompromiss zwischen einer zu hohen und niedrigen Granularität ansehen: kleinere Betrachtungen, so zum Beispiel pixelbasierte Verfahren, wären deutlich rechenintensiver und anfälliger auf Bildrauschen und würden daher keine

akkuraten Rückschlüsse bei Veränderungen des Pixelwertes ermöglichen. Bei größeren Betrachtungen wären relevante Veränderungen deutlich schwerer wahrzunehmen, speziell im DC-Wert, da dieser stets den Mittelwert aller Subpixel der Matrix darstellt.

2.5.1 DC-Wert

Da Bildaufnahmen von Mittelinfrarot-Kameras stets schwarzweiß sind (und warme Bereiche heller dargestellt werden als kühle) besitzt ein Pixel genau einen Informationskanal mit einem Grauwert zwischen 0 (schwarz) bis 255 (weiß). Der DC-Wert liegt somit als Mittelwert ebenso in diesem Intervall. Dieser Bereich ist zu groß um als Eingabealphabet für das HMM dienen zu können, da zum Beispiel der Übergang von einem Grauwert von 230 auf 235 nicht aussagekräftig ist, da es sich hierbei um Rauschen oder aber eine Neukalibrierung der Kamera, die sich ja stets auf die gesamte 8×8 Matrix auswirkt, halten kann.

Nimmt man einen kühlen (dunklen) Hintergrund, warmen (hellen) Vordergrund und ein mittelwarmen (grauen) Übergangsbereich an, so kann von einer Dreiteilung des Wärmespektrums ausgegangen werden. Diese Dreiteilung ergibt sich daraus, dass es sehr wahrscheinlich ist, dass sehr helle Bereiche zum Vordergrund und sehr dunkle Bereiche zum Hintergrund gehören. Bei dem aus verschiedenen Grauwerten bestehenden Bereich ist eine einfache Zuordnung allerdings nicht möglich. Eine feinere Einteilung des grauen Clusters ist nicht zielführend, da diese nur Zuordnungen wie *wahrscheinlich Vordergrund* oder *wahrscheinlich Hintergrund* zu ließe und somit keinen bestimmende Information darstellen würde.

Histogramm-basierte Analysen der DC-Werte über mehrere unterschiedliche Videosequenzen über alle Pixelblöcke genormt bestätigen diese Annahme und manifestieren drei Cluster, die wir vereinfacht Cluster schwarz, Cluster grau und Cluster weiß nennen - der Darstellungsfarbe des Wärmespektrums entsprechend. Die Grenzen dieser Cluster sind abhängig von jedem Video und dem betrachteten Umfeld, denn insbesondere die Kälte (und damit die Farbe) des Hintergrundes kann sich bei den jeweiligen Videos stark unterscheiden.
Wir bilden auf Basis dieser Erkenntnis die ersten 3 Symbole, wobei, wenn der DC-Wert eines Blockes in das Intervall eines Clusters fällt, wir die entsprechende Beobachtung erstellen:

- DC-Wert im Intervall Cluster schwarz → Beobachtung: DC_BLACK
- DC-Wert im Intervall Cluster grau→ Beobachtung: DC_GREY
- DC-Wert im Intervall Cluster weiß → Beobachtung DC_WHITE

Abbildung 4 zeigt eine exemplarische Einteilung in die drei Cluster. Die X-Achse beschreibt dabei den Grauwert, die Y-Achse die Anzahl Blöcke eines bestimmten Grauwerts. Für die Erstellung des Histogramms wurden sämtliche Grauwerte aller Blöcke eines Testvideos über die gesamte Länge des Videos erfasst.

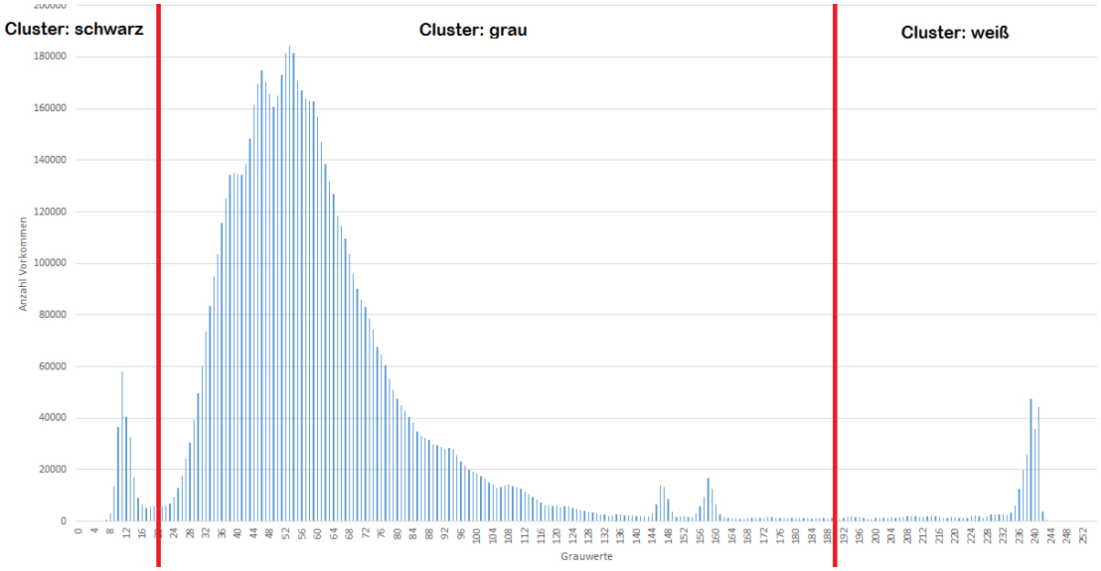

Abbildung 4: Histogramm über alle DC-Werte aller Blöcke einer Videosequenz, Clustergrenzen rot

2.5.2 AC-Wert

Die bei der Anwendung der DCT entstehende 8 x 8 Matrix enthält 63 AC-Werte. Diese bilden demnach den Großteil der in der Matrix kodierten Informationen; die Integration dieser Werte in das Eingabealphabet ist aufgrund der hohen Informationsdichte essenziell.
Eine Histogramm-basierte Betrachtung der AC-Werte analog zu der Betrachtung der DC-Werte ist nicht sinnvoll, da es sich hierbei um Offsets handelt, welche demnach im Histogramm ein Maximum um den Nullwert bilden. Wir erkennen jedoch, das Menschen aufgrund ihrer Wärmeaustrahlung eine kontrastreiche Kante zu dem Hintergrund bilden, läuft demnach ein Mensch durch einen Block, müsste eine deutliche Abweichung einiger AC-Werte zum DC-Wert entstehen. Eine Block-basierte Analyse bestätigt diese Annahme, die Standardabweichung (STD) der AC-Werte bezüglich des DC-Wertes steigt deutlich an, falls eine Person sich im Block befindet beziehungsweise durch diesen hindurch läuft.
Abbildung 5 stellt auf der X-Achse den Zeitverlauf in Frames und auf der Y-Achse den Wert der Standardabweichung für den betrachteten Bock; es lässt sich feststellen, dass die Kurve der Standabweichung stark ausschlägt, falls eine Person durch den Block läuft.

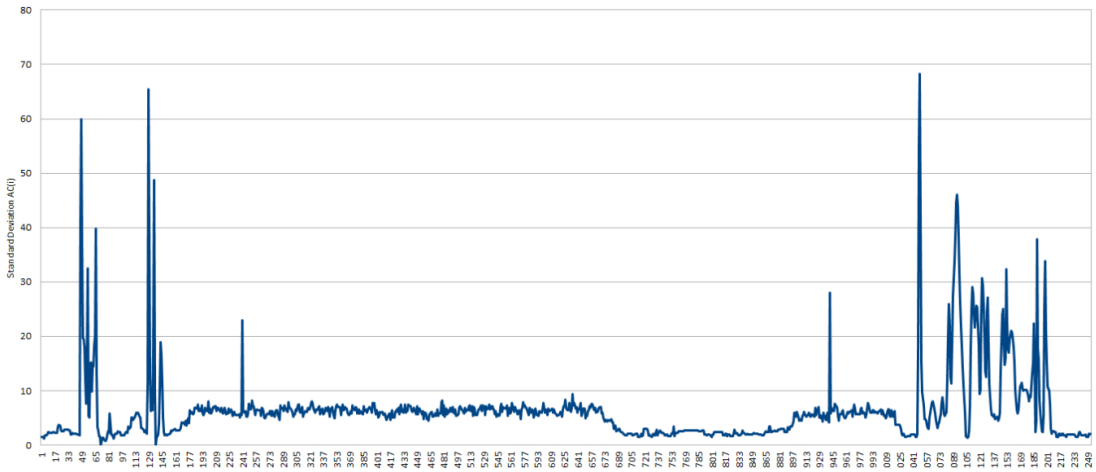

Abbildung 5: Standartabweichung der AC-Werte eines diskreten Blocks über der Zeit

Anhand von empirischen Tests lässt sich ein Schwellwert ermitteln. Falls die Standardabweichung diesen übersteigt, betritt oder verlässt mit hoher Wahrscheinlichkeit eine Person den betrachteten Block. In dem vorgestellten Beispiel liegt dieser bei einer STD von circa 25.
Zusätzlich ist zu beachten, dass die Standardabweichung von wenigen, aber extremen statistischen Ausreißern kaum beeinflusst wird, diese Situation liegt jedoch während der Erkennung von Menschen vor, insbesondere bei Personen, die Kleidung tragen, welche die Wärmeabgabe der Person dämmt und dennoch einige freiliegende Körperteile sichtbar sind. In diesem Fall ist der Großteil der AC-Werte relativ klein und der DC-Wert des betrachteten Blockes für gewöhnlich innerhalb des grauen Intervalls. Um in diesen Situationen den Vordergrund besser zu erkennen, prüfen wir mithilfe eines selbstdefinierten Verfahrens outliers() ob stark-positive (warme) Ausreißer vorliegen, welches den Wahrheitswert true zurück liefert, falls eine gewissen Anzahl von Ausreißern erkannt wird.
Vor allem bei Kleidung tragenden Personen führt dies dazu, dass nicht nur ihre Körperkanten durch die STD erkannt werden, sondern auch die Körpermitte mathematisch erfasst wird.
Infolge dieser Erkenntnise bilden wir die zwei weiteren Beobachtungen:

- (STD von AC_{1-63} < Threshold)$\wedge$!Outliers $\rightarrow$ Beobachtung AC_LOW
- (STD von AC_{1-63} > Threshold) $\vee$ Outliers $\rightarrow$ Beobachtung AC_HIGH

Auf Basis unserer Beobachtungen können nun anhand aller möglichen Permutationen von DC- und AC-Beobachtungen die Symbole unseres Eingabealphabets gebildet werden. Diees ist in Tabelle 1 dargestellt.

Alphabetsymbol	DC-Wert	Standartabweichung AC
Symbol 1	DC_BLACK	AC_LOW
Symbol 2	DC_BLACK	AC_HIGH
Symbol 3	DC_GREY	AC_LOW
Symbol 4	DC_GREY	AC_HIGH
Symbol 5	DC_WHITE	AC_LOW
Symbol 6	DC_WHITE	AC_HIGH

Tabelle 1: Eingabealphabet

2.6 Verfahren zur Vordergrundextraktion

Unser Verfahren zur Vordergrund-Extraktion basiert auf den zuvor genannten Algorithmen, der Bildpartitionierung in Blöcke und dem eingeführten Eingabealphabet. Jeder Block wird durch ein eigenes, individuelles HMM modelliert, da Blöcke unterschiedliche Übergangs-/Emissionswahrscheinlichkeiten aufgrund ihrer Lokalität aufweisen: ein Block an einer Tür ist öfter Veränderungen ausgesetzt, da erwartungsgemäß häufiger Personen durch diesen Bereich laufen, als ein Block in einem weniger frequentierten Bereich der eher dazu neigt kühl und dunkel zu sein.
In der ersten Phase ist ein Trainingsvideo zu verwenden, welches dazu dient, (individuelle) Trainingssequenzen für die Blöcke zu erstellen und im Folgenden diese dem Baum-Welch-Algorithmus zu übergeben, welcher die HMMs blocklokal trainiert. Da in einem Block hauptsächlich der Hintergrund zu beobachten ist, wird jeder Block nach der Lernphase seinen eigenen Hintergrund lernen und somit die korrespondierenden Emissionen als wahrscheinlicher bewerten.
In der zweiten Phase kann bereits ein Live-Video (in der Anwendung) eingesetzt werden, das HMM ist in dieser Phase zeitinvariant.

Für jeden Block wird die aktuelle Beobachtung gebildet. Die HMMs werden nun zur Vorhersage verwendet: Unter Verwendung des Forward- oder Viterbi-Algorithmus kann nun die nächste Emission prognostiziert werden. Diese wird mit der tatsächlich vorliegenden Emission (eine der 6 definierten Symbole unseres Eingabealphabetes) verglichen. Bei Gleichheit handelt es sich offensichtlich um den erlernten Hintergrund, bei Abweichungen ist ein Vordergrundobjekt in den Block eingetreten. Die aktuelle Emission wird an eine Warteschlange der k-letzten Emissionen angehangen und die Operationen wiederholt.

3 Implementierung und praktische Details

Dieses Kapitel gibt einen kurzen Überblick zu den von uns verwendeten Technologien, die wir zur Realisierung des theoretischen Entwurfs aus Kapitel 2 verwendet haben.

3.1 Videomaterial

Das uns zur Verfügung gestellte Bildmaterial besteht aus einzelnen Videosequenzen, wir können also nicht auf Live-Material arbeiten.

In der Praxis spielt dies jedoch keine große Rolle, da unter Linux der Zugriff auf eine Videodatei sich nicht wesentlich von dem Zugriff auf eine Kamera unterscheidet. Insgesamt standen uns sieben unterschiedliche Videosequenzen zur Verfügung, von denen sechs paarweise entstandene Aufnahmen sind. Das bedeutet, dass Kameraposition sowie Winkel zwischen den beiden Videosequenzen eines Paares nicht differiert, lediglich die aufgenommenen Szenen sind unterschiedlich. Diese Tatsache ist insofern hilfreich, als das wir die Möglichkeit haben, auf einer Videosequenz zu lernen und die erlernten Parameter später auf der anderen Videosequenz anzuwenden.
Der größte Nachteil der Verwendung von Videosequenzen ist, dass die einzelnen Sequenzen relativ kurz sind (Dauer übersteigt nicht 10 Minuten) und es somit nicht möglich ist, wirklich lange Lernphasen von zum Beispiel einigen Stunden zu realisieren. Der Vorteil ist hingegen, dass man Veränderungen an Programmbestandteilen und Parametern immer wieder an den selben Sequenzen ausprobieren kann und somit den entstehenden Effekt besser nachvollziehen kann.
Sämtliche Videosequenzen laufen mit 25 Frames pro Sekunde, was auf jeden Fall genug Daten zur Auswertung liefert. In der Praxis genügen vermutlich auch schon weniger Frames, da anhand der vorliegenden Videos deutlich wurde, dass sich die Personen im Bild nicht so schnell bewegen, dass sich innerhalb eines Bruchteils einer Sekunde die Szenerie stark verändert.

3.2 C++

Die Implementierung sowohl unseres Testbeds als auch die finale Implementierung als Komponente zur Vordergrundextraktion in einem vorgegebenen Framework fand in C++ statt. Für die Wahl von C++ gab es verschiedene Gründe, von denen vor allem die Performance, die Portabilität sowie das gute Angebot an zur Verfügung stehenden Bibliotheken im Vordergrund standen.

3.3 OpenCV

OpenCV ist eine Bibliothek die eine Vielzahl unterschiedlicher Algorithmen für die Bildbearbeitung und somit letztlich für die Videobearbeitung zur Verfügung stellt. Hinzu kommt, dass auch Funktionen für das Lesen, Schreiben und Abspielen von Videodateien verschiedener Datentypen sowie von Kamera angeboten werden. Videos und Bilder werden in das OpenCV spezifische Format Mat eingelesen, auf dem viele Operationen möglich sind. Die von uns vorgestellte DCT ist Teil dieser Bibliothek und konnte von uns direkt auf den Mat-Objekten angewandt werden.
OpenCV bietet Interfaces für C, Python, Java und C++ und lässt sich auf vielen unterschiedlichen Plattformen einsetzen.

3.4 CvHMM

Aus der Menge der vorhandenen C++-HMM Bibliotheken fiel unsere Wahl auf CvHMM[7], vor allem da diese Bibliothek direkt auf dem in OpenCV enthaltenen Standard-Videodatentyp Mat operiert und daher die Vermutung nahe liegt, dass weniger Performanceeinbußen vorliegen, da keine Konvertierung von Daten in andere Formate erfolgen muss.

4 Evaluation

Am Ende unserer Arbeit wurde uns ein Framework des SAFEST-Projekts zur Verfügung gestellt, dass die Möglichkeit bietet, verschiedene vorimplementierte Algorithmen mit einander zu vergleichen.
Nachdem wir unseren Algorithmus in das Framework integriert hatten, haben die folgenden Vergleiche mit einer vorimplementierten, Histogramm-basierten Lösung angestellt:

a) die vorimplementierte Lösung, die mit Histogrammen arbeitet

b) unsere HMM-basierte Lösung mit generischen Werten für Initialwerte, Emission und Transition ohne Lernen

c) unsere HMM-basierte Lösung mit generischen Werten für Initialwerte, Emission und Transition mit Lernen

Weitere Vergleiche haben wir nicht angestellt, da das Histogramm-basierte Verfahren den anderen Verfahren entweder überlegen ist (OpenCV MOG und OpenCV MOG2) oder aber das Verfahren den Hintergrund hart auf bestimmte Videos kodiert (MEAN) und sehr unflexibel ist, sobald der Hintergrund nicht mehr uniform ist.
Das Ergebnis haben wir jeweils visualisiert, indem wir für jeden zu evaluierenden Algorithmus die Abweichung vom eigentlich erwarteten Wert visualisiert haben. Ein optimal funktionierender Algorithmus würde also als Bild im Diagramm ohne Abweichungen die Nulllinie verdecken.
Die erwarteten Werte für diesen Vergleich sind uns seitens des SAFEST Projekts zur Verfüngung gestellt worden.

4.1 Vergleich mit Histogramm-basierter Implementierung: Innenhof

In Grafik 6 sieht man, dass der Histogramm-basierte Algorithmus für dieses Video bessere Ergebnisse liefert, als der HMM-basierte Algorithmus.

4.2 Vergleich mit Histogramm-basierter Implementierung: Tegel

Wie man Abbildung 7 entnehmen kann, sind die Ergebnisse beider Algorithen vom Verlauf her sehr ähnlich. Allerdings hat der HMM-basierte Algorithmus gegenüber dem

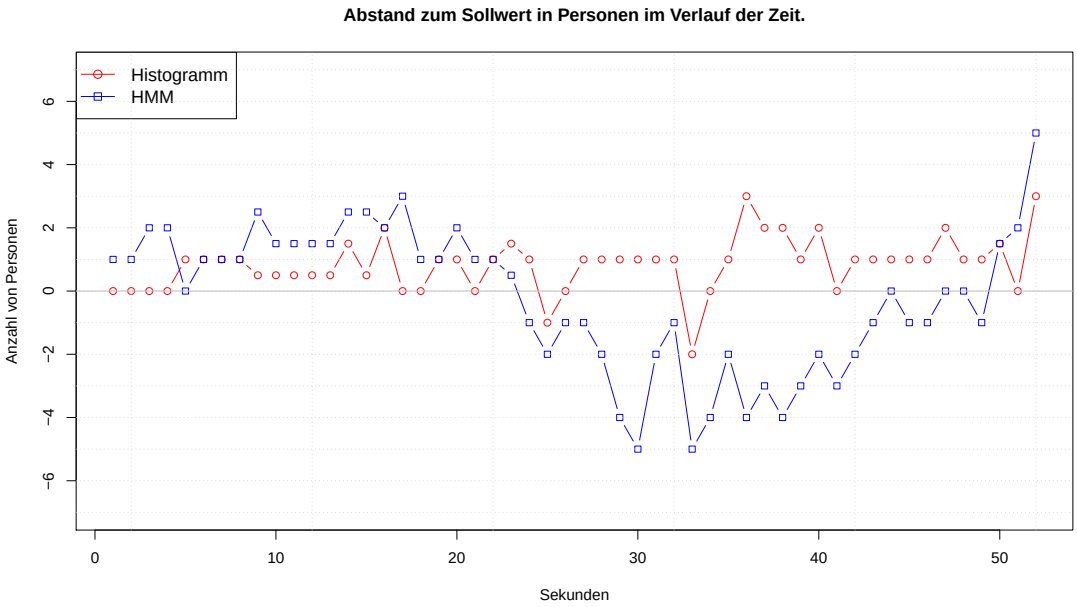

Abbildung 6: Innenhof: Histogramm vs. trainiertes HMM

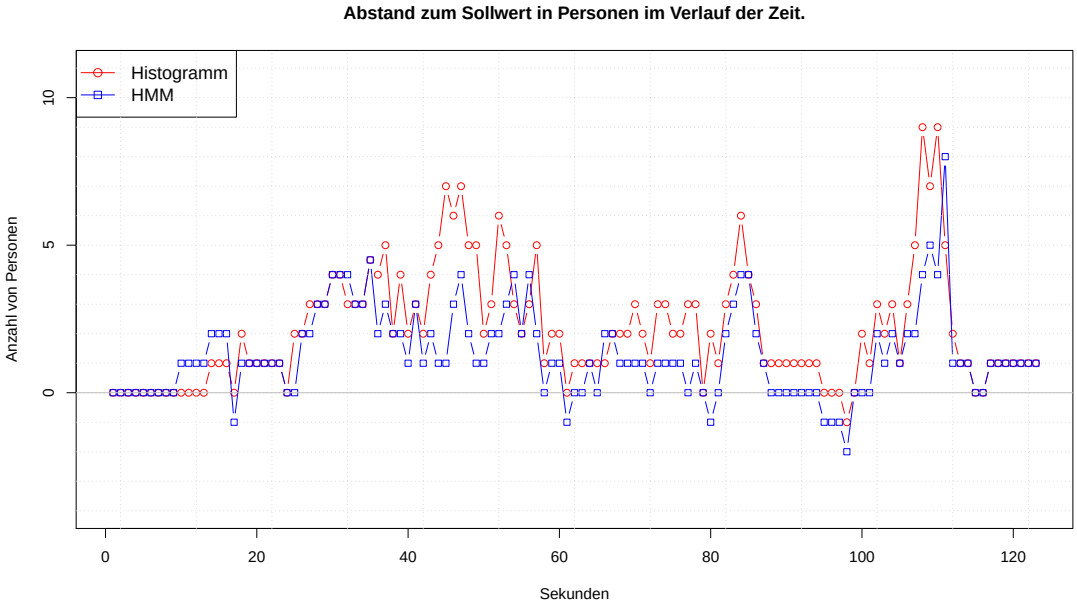

Abbildung 7: Tegel: Histogramm vs. trainiertes HMM

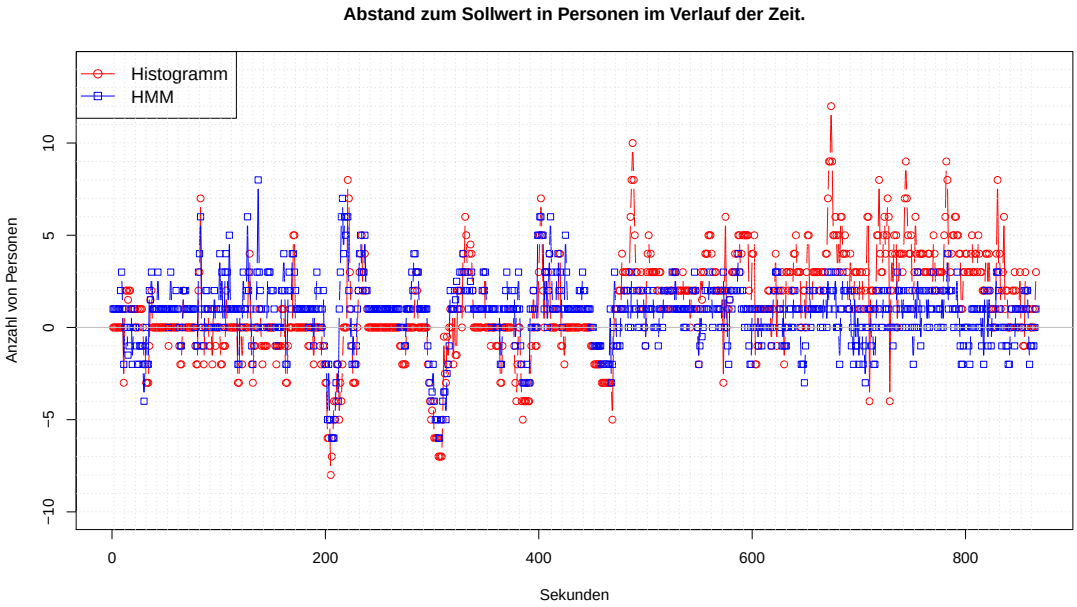

Abbildung 8: Eingang: Histogramm vs. trainiertes HMM (gesamt)

Histogram-basierten kleine Vorteile. Zwischen 30 und 50 Sekunden hat er eine deutlich geringere Abweichung vom korrekten Wert. Dasselbe gilt für den Bereich zwischen ca. 65 und 80 Sekunden.

4.3 Vergleich mit Histogramm-basierter Implementierung: Eingang

In der Grafik 8 sieht man, dass beide Algorithmen hier nicht wirklich gut funktioieren. Dies ist aufgrund der Eigenschaften des Videos zu erwarten: der Hintergrund ist sehr hell und die Kalibrierung der Kamera fällt hier auch optisch sehr stark auf.
Während in der ersten Hälfte des Videos der Histogramm-basierte Algorithmus noch Vorteile hat (relativ viele korrekte Anzeigen, der HMM-basierte Algorithmus erkennt meist eine Person zu viel). ist der HMM-basierte Algorithmus in der zweiten Hälfte des Videos (ab etwa Frame 450) deutlich besser. in Grafik 9 sieht, gibt es wesentlich weniger Ausreißer und das Ergebnis ist insgesamt auch viel dichter am korrekten Wert.

4.4 Diskussion

Aus den vorangegangenen Grafiken kann man erst einmal ablesen, dass beide Algorithmen grundsätzlich dazu geeignet sind, Vorder- von Hintergrund zu trennen. Da die Graphen je einmal für den Histogramm-basierten und den HMM-basierten Algorithmus

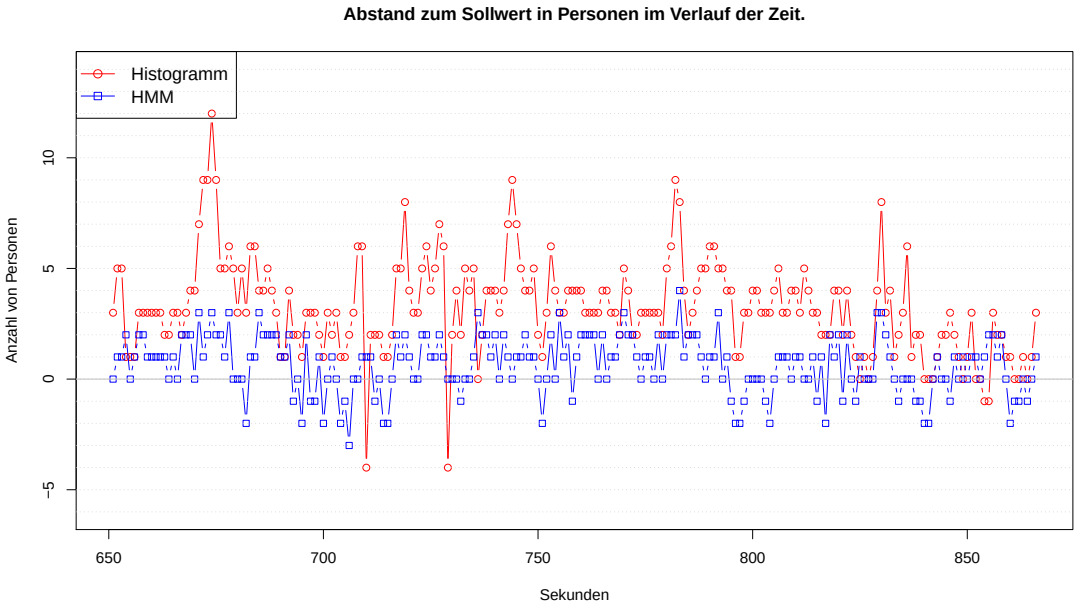

Abbildung 9: Eingang: Histogramm vs. trainiertes HMM, Auschnitt Frame 651-Ende

darstellen, fallen einige Details auf, die erklärt werden müssen. Außer im Falle von Grafik 6 (Innenhof) sind die Verläufe der beiden Graphen sehr ähnlich. Dies legt nahe, dass der dargestellte Fehler nicht nur an den geprüften Algorithmen liegt, sondern jedenfall teilweise durch das Clustering erzeugt wird. Dieser Umstand deckt sich mit der Beobachtung, dass bei den Tegel- und Eingang-Videos einzelne Personen in zwei oder mehr Cluster geteilt werden.
Wie aus Grafik 8 und 9 (Eingang) hervorgeht, hat der von uns entwickelte HMM-basierte Algorithmus besonders bei schwieriger zu verarbeitendem Videomaterial Vorteile bei der Erkennung.
Allerdings haben wir nur „Offline-Learning“genutzt, da die uns zur Verfügung gestellten Sequenzen für „Online-Learning“zu kurz waren. Wenn wir versucht haben, online zu lernen, führte dies immer zu schlechteren Ergebnissen aufgrund von Geisterbildern.
Den größten Vorteil hat der Histogramm-basierte Algorithmus bei der Performance: whärend hier mit 22-25 fps eine flüssige Videodarstellung noch möglich war, lief der HMM-basierte Algorithmus nur mit ca. 10-12 fps. Hier wäre also noch Potential zur Verbesserung.

4.5 CvHMM

Leider hat sich im Laufe des Projekts gezeigt, dass CvHMM leider sehr ineffizient auf einzelne Daten zugreift. Dies konnten wir mit Hilfe von Callgrind (Teil von Valgrind)

nachweisen. Hieraus ergeben sich auch die in 4.4 angesprochenen Performance Probleme. Ein weiterer großer Nachteil von CvHMM ist, dass nur der Baum-Welch-, der Viterbi- und der Decode-Algorithmus angeboten werden. Gerade hier ist wiederum ein Manko zu sehen, da wir später den Forward-Algorithmus gebraucht hätten und nun stattdessen den Viterbi-Algorithmus verwenden müssen, der jedoch ineffizienter als der Forward-Algorithmus ist.
Zusammenfassend lässt sich zur Wahl der HMM Bibliothek sagen, dass es wohl das Ergebnis des Projekts erheblich hätte verbessern können, wenn unsere Wahl auf eine andere Bibliothek gefallen wäre.

5 Fazit und Ausblick

Wir werten das Ergebnis unseres Projekts als Erfolg.
Wir haben nachgewiesen, dass es möglich ist, einen auf HMM-basierenden Algorithmus zur Trennung von Vorder- und Hintergrund zu verwenden und damit zumindest ähnliche Ergebnisse zu erzielen, wie min den bereits implementierten Algorithmen.
Gerade bei für die Trennung schwierigen Bilddaten, wie beispielsweise den „Eingang“Videos hat unsere Implementierung dabei klare Vorteile.
Die in den Abschitten 4.4 und 3.4 angesprochenen Nachteile können sehr wahrscheinlich auch noch weiter verbessert werden.
Vorschläge für Verbesserungen wären zum einen die Verwendung einer besseren Bibliothek für HMMs sowie einige algorithmische Anpassungen.
Mögliche Kandidaten für eine bessere HMM-Bibliothek wären dabei HMMlib oder vielleicht sogar die, wie wir später herausgefunden haben, in OpenCV enthaltenen HMM-Funktionen, die allerdings innerhalb des OpenCV-Projekts nicht weiterentwickelt werden und derzeit des Status “deprecated” haben. Gerade HMMlib ist sehr viel mehr auf Performance optimiert, setzt dafür aber auch stärkere Hardware vorraus (SSE3).
Die Verwendung einer anderen HMM-Bibliothek würde auch die Chance eröffnen, statt den doch relativ aufwändigen Viterbi-Algorithmus zur Bestimmung des nächsten Zustands den Forward-Algorithmus zu verwenden. Dies würde vermutlich auch zu einer Performance Steigerung führen.
Zusammenfassend kann man aber sagen, dass der Einsatz eines HMM-basierten Hintergrund-Extraktors sinnvoll erscheint, da er auch mit schwierigen Videos gut umgehen kann. Interessant wäre hier noch, ob es möglich ist, eine sinnvolle online-Lernfähigkeit zu integrieren, wenn die verwendeten Videosequenzen lang genug sind (was im Falle von realen Überwachungskameras ja kein Problem darstellen würde).

Abbildungsverzeichnis

Literatur

[1] N. Dalal und B. Triggs, "Histograms of oriented gradients for human detection", in *In CVPR*, 2005, S. 886–893.

[2] S. Dasgupta, "Learning mixtures of gaussians", in *FOCS*, 1999, S. 634–644.

[3] M. Stamp, *A revealing introduction to hidden markov models*, 2004.

[4] S. A. Khayam, *The discrete cosine transform (dct): theory and application". department of electrical & computing engineering*, 2003.

[5] M. Gales und S. Young, "The application of hidden markov models in speech recognition", *Found. Trends Signal Process.*, Bd. 1, Nr. 3, S. 195–304, Jan. 2007, ISSN: 1932-8346. DOI: 10.1561/2000000004. Adresse: http://dx.doi.org/10.1561/2000000004.

[6] L. Yang, B. Widjaja und R. Prasad, "Application of hidden markov models for signature verification", *Pattern Recognition*, Bd. 28, Nr. 2, S. 161–170, 1995, ISSN: 0031-3203. DOI: http://dx.doi.org/10.1016/0031-3203(94)00092-Z. Adresse: http://www.sciencedirect.com/science/article/pii/003132039400092Z.

[7] O. Sakhi, *Cvhmm - discrete hidden markov models based on opencv*. Adresse: http://sourceforge.net/projects/cvhmm/,%20Sourceforge%20on%202012-06-10.

[8] M. Lamarre, *Tracking and Activity Classification in Video Surveillance Applications*. McGill University, 2002. Adresse: http://digitool.library.mcgill.ca/webclient/StreamGate?folder_id=0&dvs=1393296165809~863.