Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik VI
Prof. Dr.-Ing. Hermann Ney

Studienarbeit

Erweiterung eines holistischen statistischen Bilderkenners zur Verwendung von mehreren Merkmalen

David Rybach

Matrikelnummer 229 337

März 2005

Betreuer: Dipl.-Inform. Daniel Keysers

Inhaltsverzeichnis

Zusammenfassung

Statistische Bilderkenner, die mit einem holistischen Ansatz arbeiten, verwenden Wahrscheinlichkeitsverteilungen als Modelle für Objekte und deren Hintergründe. Die Modelle werden aus ortsabhängigen Merkmalen der Trainingsbilder berechnet, z.B. aus den Grauwerten. In dieser Arbeit wird ein Verfahren vorgestellt, das mehrere Merkmale in die statistischen Modelle integriert. Dadurch konnten die Fehlerraten auf zwei verschiedenen Datensätzen verringert werden.

1 Einleitung

Der erste Schritt in den meisten Verfahren zur Bilderkennung ist die Berechnung von Merkmalen aus den Bilddaten. Dies können ganz einfache Merkmale sein, zum Beispiel die Grau- oder Farbwerte des Bildes, oder Daten, die aus Berechnungen auf dem Bildmaterial hervorgehen. Die Merkmalsdaten werden analysiert und vom Klassifikator benutzt. Welche Merkmale zur weiteren Verarbeitung genutzt werden, hängt von der Aufgabenstellung und vom zu untersuchenden Bildmaterial ab.

Bilderkenner, die mit einem holistischen statistischen Verfahren arbeiten, können komplexe Szenen analysieren und die abgebildeten Objekte klassifizieren. Sie „erklären“ das gesamte Bild mit statistischen Modellen. Es ist also notwendig, Modelle für Objekte und Hintergründe zu verwenden. Die Bestimmung der Position und Größe des Objekts und dessen Klassifizierung geschehen in einem Schritt. Viele andere Verfahren müssen vor der Klassifikation eine Segmentierung des Bildes durchführen, die alleine schon fehleranfällig ist. Auch für das automatische Objekttraining werden unsegmentierte Daten benutzt, für die nur die Klasse des abgebildeten Objekts bekannt ist. Daher müssen die Methoden, die zur Klassifikation genutzt werden, auch für das Training angewendet werden. Die Verwendung von unsegmentierten Daten ist wünschenswert, um die manuelle Arbeit an den Daten zu minimieren.

In dieser Studienarbeit wird die Erweiterung eines holistischen Bilderkenners vorgestellt, die es ermöglicht, mehrere Merkmale eines Bildes zu verarbeiten. Um den Einfluss von unterschiedlichen Merkmalen und von Zusammenstellungen von Merkmalen auf die Erkennungsleistung zu untersuchen, wurde das System so gestaltet, dass Merkmale einfach auswählbar und kombinierbar sind. Neben einfachen Grau- und Farbwert-Merkmalen wurden auch komplexere Merkmale implementiert. Um die Merkmale eines Bildes zu kombinieren, wird ein neues „Bild“ berechnet, das aus mehreren Schichten besteht. Die einzelnen Schichten repräsentieren Merkmale oder Teile eines Merkmals. Zur Berechnung der Ähnlichkeit zwischen einem Modell, das ebenfalls aus mehreren Schichten besteht, und einem Bild werden die Schichten in unterschiedlicher Gewichtung berücksichtigt.

Je mehr Merkmale eines Bildes verwendet werden, desto mehr Daten stehen zur Berechnung der Modelle zur Verfügung. Es wird erwartet, dass sich die Verwendung mehrerer Merkmale positiv auf die Erkennungsleistung auswirkt. In Abschnitt 5 wird diskutiert, ob sich diese Erwartung bestätigt.

In den folgenden Abschnitten werden zunächst die Grundlagen des Verfahrens erklärt. Anschließend wird der Aufbau und die Implementierung des gesamten Systems und der einzelnen Merkmalsberechnungen vorgestellt. Abschnitt 4 dokumentiert die durchgeführten Experimente. Außerdem werden dort die untersuchten Bilddatenbanken beschrieben. Im Anhang befindet sich die Dokumentation zur erstellten Software.

2 Verfahren

2.1 Grundlagen

Ausgangspunkt dieser Studienarbeit ist die Software, die von M. Motter im Rahmen seiner Diplomarbeit [1] entwickelt wurde. Die theoretischen Grundlagen werden in [1] detailliert erläutert und sollen in dieser Arbeit nur kurz vorgestellt werden.

Das Verfahren basiert auf einem statistischen Ansatz zur Objekterkennung mit einem ganzheitlichen Modell. Dabei werden statistische Modelle für das Objekt und den Hintergrund verwendet, um alle Pixel eines Bildes zu erklären. Das Objekt wird als quadratischer Ausschnitt des Bildes angenommen. Die Grauwerte der Pixel aus diesem Ausschnitt bilden den Merkmalsvektor $\mathbf{x}$ für das Objekt. Alle Pixel, die außerhalb des Ausschnitts liegen, werden dem Hintergrund zugeordnet.

Als Modell für Objekte wird eine Gauß'sche Mischverteilung verwendet, mit der die klassenbedingte Wahrscheinlichkeitsverteilung $p(\mathbf{x}|k)$ beschrieben wird.

$$p(\mathbf{x}|k) = \sum_{j=1}^{J_k} c_{jk}\, \mathcal{N}\left(\mathbf{x}|\mu_{jk}, \sigma_k^2\mathbf{I}\right) \qquad \text{mit} \qquad \sum_{j=1}^{J_k} c_{jk} = 1 \quad , \quad c_{jk} \geq 0 \tag{1}$$

$p(\mathbf{x}|k)$ ist die klassenbedingte Wahrscheinlichkeitsfunktion für einen Merkmalsvektor $\mathbf{x}$ und eine Klasse k mit Gewichten c_{jk} für die Dichten $\mathcal{N}(\mathbf{x}|\mu_{jk}, \sigma_k^2\mathbf{I})$. Die einzelnen Dichten sind multivariate Gauß-Verteilungen mit Parametern μ_k und Σ_k. Es wird eine über alle Komponenten gepoolte Varianz verwendet, so dass $\Sigma_k = \sigma_k^2\mathbf{I}$ gesetzt wird.

$$\mathcal{N}\left(\mathbf{x}|\mu_k, \sigma_k^2\mathbf{I}\right) = \frac{1}{\sqrt{\left(2\pi\sigma_k^2\right)^D}} \cdot \exp\left(-\frac{1}{2\sigma_k^2} \cdot \|\mathbf{x} - \mu_k\|^2\right) \tag{2}$$

Die Menge von Pixeln $\mathbf{X}_{S_0}$, die nicht zum Objekt gehören, werden mit einer univariaten Gaußverteilung, dem Hintergrundmodell, beschrieben.

$$p(\mathbf{X}_{S_0}|\mu_0, \sigma_0^2) = \prod_{x \in \mathbf{X}_{S_0}} \frac{1}{\sqrt{2\pi\sigma_0^2}} \cdot \exp\left(-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right) \tag{3}$$

Für die Klassifikation eines Bildes werden verschiedene Transformationen (Translation und Skalierung) des Objekts betrachtet. Sei $p(\mathbf{X}_{S_1}|\mu_k)$ die Wahrscheinlichkeit, dass der Bildausschnitt $\mathbf{X}_{S_1} = \{X_{xy} : (x,y) \in S_1\}$ vom Modell für die Klasse k erzeugt wird. Und sei $p(\mathbf{X}_{S_0}|\mu_0)$ die Modellannahme für den Hintergrund. Da für alle Klassen k und alle Transformationen ϑ die gleichen Wahrscheinlichkeiten $p(k)$ bzw. $p(\vartheta)$ angenommen werden, ergibt sich folgender Klassifikator für ein Bild $\mathbf{X}$:

$$r(\mathbf{X}) = \underset{k}{\operatorname{argmax}} \left\{ \max_{\vartheta} \left\{ p(\mathbf{X}_{S_0}|\mu_0) \cdot p(\mathbf{X}_{S_1}|\mu_k(\vartheta)) \right\} \right\} \tag{4}$$

wobei die Partitionierung des Bildes in S_1 und S_0 durch die Transformation ϑ bestimmt wird.

Die Modelle werden in einem automatischen Objekttraining berechnet. Im Training liegen keine vorsegmentierten Daten vor, sondern komplexe Szenen aus Objekt und Hintergrund. Der Trainingsalgorithmus muss daher die Positionen und Skalierungen von Objekten auf den Trainingsbildern bestimmen, um Modelle für das Objekt und den Hintergrund zu berechnen. Dazu wird eine abgewandelte Form der Objektklassifizierung verwendet, die im Bild nach der Hypothese (Position und Skalierung) sucht, die am besten zum Referenzmodell passt:

$$\hat{\vartheta} = \underset{\vartheta}{\operatorname{argmax}} \left\{ p(\mathbf{X}_{S_0}|\mu_0) \cdot p(\mathbf{X}_{S_1}|\mu_1(\vartheta)) \right\} \tag{5}$$

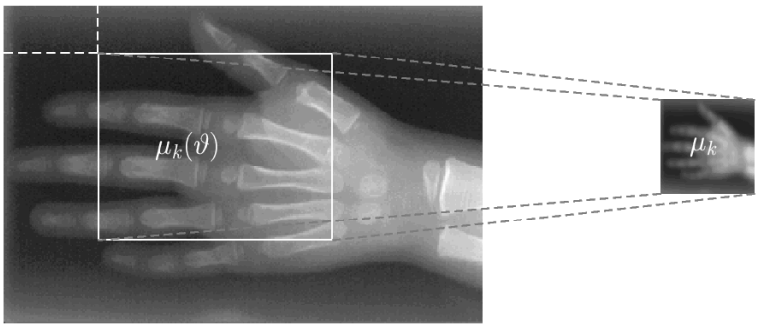

Abbildung 1: Projektion des Referenzmodells auf ein Trainingsbild (aus [1])

Abbildung 1 zeigt, wie die Transformation $\mu_k(\vartheta)$ des Referenzmodells μ_k auf das Trainingsbild projiziert wird.

Im Trainingsalgorithmus wird für ein Trainingsbild einer bekannten Klasse die beste Transformation gesucht. Dies geschieht in einem iterativen Verfahren, in dem, nach der Berechnung eines initialen Modells, Hypothesen für die Trainingsbilder berechnet werden und damit ein neues Referenzmodell. Mit den neuen Modellen werden wieder Hypothesen berechnet, so dass sich die Modelle immer weiter verbessern. Als Startwerte können Konstanten für Mittelwert und Varianz des Objekts und des Hintergrunds angegeben werden. Alternativ kann für das initiale Referenzmodell des Objekts auch ein Bild aus den Trainingsdaten verwendet werden. Zur Berechnung der besten Hypothese werden alle Transformationen auf das Referenzobjekt angewendet und mit dem Trainingsbild verglichen. Aus den Hypothesen werden mit dem EM-Algorithmus die Parameter für die Modelle geschätzt.

Das Matching von Referenzmodellen und Bildern, sowie Training und Klassifikation werden in den Abschnitten 2.3, 2.4 und 2.5 genauer beschrieben.

2.2 Erweiterung

Die in Abschnitt 2.1 beschriebene Software wurde dahingehend erweitert, dass neben den Grauwerten eines Bildes auch andere Merkmale (*Features*) verwendet werden können. Dazu wird aus einem Bild eine Sammlung von Merkmalen berechnet.

Für die formale Beschreibung betrachtet man ein Bild der Größe $X \times Y$ als Funktion $g : X \times Y \to \mathbb{R}$, die jedem Punkt des Bildes einen Grauwert zuordnet. Ein Feature $f_i : X \times Y \to \mathbb{R}^{n_i}$ ordnet jedem Pixel einen Vektor von n_i Werten zu. Für ein Feature „Farbe“ kann dies zum Beispiel der Rot-, Grün- und Blauanteil sein. Mehrere Features eines Bild können dann als Funktion $f : X \times Y \to \mathbb{R}^D$ beschreiben werden:

$$f(x,y) = \begin{pmatrix} v_{1,1} \\ \dots \\ v_{1,n_1} \\ \dots \\ v_{i,1} \\ \dots \\ v_{i,n_i} \\ \dots \\ v_{m,n_m} \end{pmatrix} \quad \text{mit} \quad f_i(x,y) = \begin{pmatrix} v_{i,1} \\ \dots \\ v_{i,n_i} \end{pmatrix}, \quad \sum_{i=1}^{m} n_i = D \tag{6}$$

Es entsteht ein neues Bild aus mehreren Schichten (*Layern*), das in Abbildung 2 schematisch

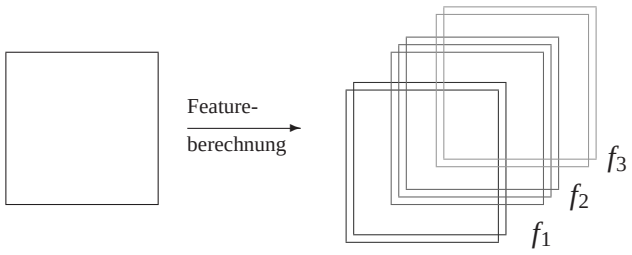

Abbildung 2: Featureberechnung erzeugt ein „Multi-Layer Bild“

dargestellt ist. Im folgenden wird für Bild $\mathbf{X} \in \mathbb{R}^{X \times Y}$ eine Sammlung von Features $\mathbf{F} \in \mathbb{R}^{D \times X \times Y}$ angenommen, in der ein Punkt (x,y) im Layer d mit $\mathbf{F}[d,x,y]$ referenziert wird. Die Referenzmodelle (*Templates*) werden mit $\mathbf{T} := \mu(\vartheta) \in \mathbb{R}^{D \times X_t \times Y_t}$ ($X_t \leq X$, $Y_t \leq Y$) bezeichnet.

Um mehrere Features im Matching verwenden zu können, wurde die Berechnung der quadratischen euklidischen Distanz $\|\mathbf{X} - \mu(\vartheta)\|^2$ (aus Formel 2) zwischen einem Bild $\mathbf{X}$ und einem Referenzmodell $\mu(\vartheta)$ angepasst. Für eine Sammlung von Features $\mathbf{F}$ und ein Template $\mathbf{T}$ ist die Distanz für eine Position (x,y) des Templates und eine feste Skalierung

$$D(x,y) = \sum_{d=1}^{D} \left(\frac{1}{D} \cdot \sum_{x'=x}^{X_t+x} \sum_{y'=y}^{Y_t+y} \left(\mathbf{F}[d,x',y'] - \mathbf{T}[d,x'-x,y'-y] \right)^2 \right) \tag{7}$$

Außerhalb des Definitionsbereichs des Bildes bzw. des Templates werden $\mathbf{F}$ und $\mathbf{T}$ als 0 definiert.

Um den Einfluss der Features auf die Distanz zu steuern, wurden Gewichte w_d für die Layer eingeführt. Unter Verwendung dieser Gewichte ergibt sich die Distanz als

$$D(x,y) = \sum_{d=1}^{D} \left(w_d \cdot \sum_{x'=x}^{X_t+x} \sum_{y'=y}^{Y_t+y} \left(\mathbf{F}[d,x',y'] - \mathbf{T}[d,x'-x,y'-y] \right)^2 \right) \quad \text{mit} \quad \sum_{d=1}^{D} w_d = 1 \tag{8}$$

Um die Berechnung der Distanzen effizienter zu gestalten, wird in [1] die Distanzberechnung im Frequenzraum durchgeführt und zusätzlich eine effiziente Berechnung der Quadratsummen verwendet. Diese Verfahren wurden für die Verwendung mehrere Features erweitert und in der erstellten Software benutzt. Die Erweiterung wurde analog zu Formel 8 durchgeführt und soll hier nicht weiter beschrieben werden.

Weiterhin wurde der EM-Algorithmus ([1, Anhang A], [2]) erweitert, um die Qualität der erzeugten Dichten zu verbessern. Dazu wurde ein Grenzwert für die Varianz der Pixel des Mittelwerts einer Dichte eingeführt. Dieser so genannte *Mean-Variance-Treshold* verhindert, dass Dichten erzeugt werden, die wenig über die Struktur des Bildes aussagen. Unterschreitet die Varianz diesen Treshold, wird die Dichte aus der Verteilung entfernt.

2.3 Matching

Im Matching werden für eine Sammlung von Features eines Bildes, im folgenden Beobachtung genannt, eine Menge von Hypothesen gesucht, die am besten zum gegeben Objekt- und Hintergrundmodell passen. Für alle Dichten des Objektmodells werden aus den Mittelwerten unterschied-

Abbildung 3: Matching eines Referenzmodells und einer Beobachtung: Die Referenz (links) wird skaliert (mitte) und die beste Position innerhalb der Beobachtung wird bestimmt (rechts).

lich skalierte Templates berechnet. Dann werden für verschiedene Positionen dieser Referenz-Templates die Distanzen zur Beobachtung berechnet. Mit den berechneten Distanzen und den Formeln aus den Abschnitten 2.1 und 2.2 werden die Templates mit der höchsten Wahrscheinlichkeit berechnet. Abbildung 3 zeigt, wie ein Objektmodell auf eine Beobachtung projiziert wird.

Um die Suche effizienter zu gestalten, wird zunächst eine Version der Beobachtung mit geringerer Auflösung betrachtet. Die besten Hypothesen, die das Matching für diese Version liefert, werden im zweiten Schritt weiterverfolgt. Im zweiten Schritt wird die Beobachtung in voller Auflösung verwendet. Für das zweite Matching werden feinere Skalierungsstufen und kleinere Verschiebungen durchsucht, ausgehend von den Werten aus dem ersten Schritt.

2.4 Training

Im Training werden aus einer Menge von Trainingsbildern mit bekannten Klassen die Gauß'schen Mischverteilungen für die Objektmodelle und Gaußverteilungen für die Hintergrundmodelle berechnet. Das Training wird nicht diskriminativ durchgeführt, sondern für jede Klasse separat.

Ausgehend von einem initialen Modell, das aus einer gegebenen Sammlung von Features und einer gegebenen Varianz berechnet werden kann, werden iterativ Verteilungen berechnet, die die Trainingsdaten erklären. In jeder Iteration wird, wie in Abschnitt 2.3 beschrieben, jedes Trainingsdatum (Sammlung von Features, die aus einem Trainingsbild berechnet wurde) mit dem aktuellen Modell gematcht. Nachdem für alle Trainingsdaten das Matching durchgeführt wurde, wird aus den besten berechneten Hypothesen mit dem EM-Algorithmus eine neues Objektmodell berechnet. Dieses wird dann in der nächsten Iteration verwendet. Diese Schritte werden solange iteriert, bis entweder die Distanz der Trainingsdaten zu den Modellen konvergiert oder eine maximale Anzahl von Iteration durchgeführt wurde. Die berechneten Verteilungen werden als Modell für die jeweilige Klasse für die Objektklassifikation verwendet. Abbildung 4 zeigt einen Teil der Trainingsbilder für eine Klasse und das daraus berechnete Objektmodell,

2.5 Klassifikation

Zur Klassifikation eines Bildes wird die aus dem Bild berechnete Sammlung von Features verwendet. Diese Featuresammlung wird mit den Modellen gematcht, die im Training für die einzelnen Klassen erstellt wurden. Der Klassifikator ordnet dann dem zu klassifizierenden Bild diejenige Klasse zu, deren Modell die geringste Distanz zum Bild hat (siehe Formel 4). Die minimale Distanz eines Modells zum Bild impliziert die maximale Wahrscheinlichkeit, dass Objekt und Hintergrund vom Modell (genauer: von den Modellen für Objekt und Hintergrund) erzeugt werden [3].

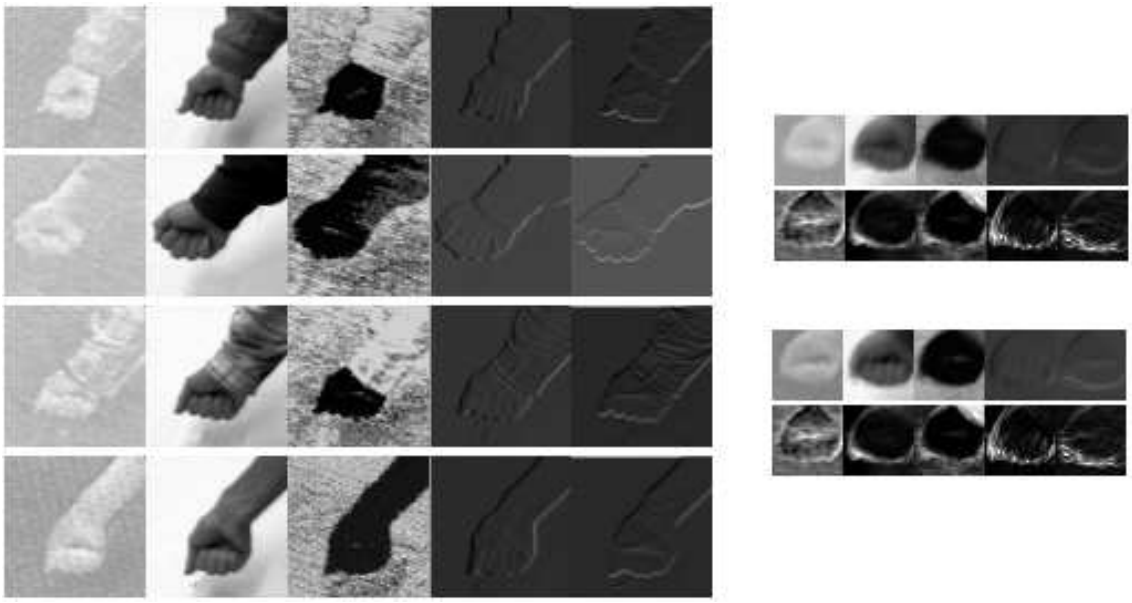

Abbildung 4: 4 von 18 Trainingsdaten (links) und das daraus berechnete Modell (rechts) mit 2 Dichten (jeweils oben der Mittelwert und darunter die Varianz)

3 Implementierung

Ausgangspunkt für die Implementation ist die holistisch statistische Bilderkennungssoftware, die von M. Motter im Rahmen seiner Diplomarbeit [1] entwickelt und vom Lehrstuhl für Informatik VI weiterentwickelt wurde. Die Implementierung wurde in der Programmiersprache C++ durchgeführt. Für die Matrix- und Vektorrechnung, die Berechnung der FFT und die Ein- und Ausgabe von Bilddateien werden externe Bibliotheken benutzt (siehe Anhang A).

3.1 Implementierte Features

Die im folgenden beschriebenen Features wurden im Rahmen dieser Studienarbeit implementiert. Die Software-Architektur, die dafür verwendet wurde, wird im nächten Abschnitt erläutert. Jede Feature-Klasse berechnet aus einer Bilddatei einen oder mehrere Layer. Alle Features berechnen Layer, die die gleiche Größe wie das ursprüngliche Bild haben. Der Wertebereich aller Features liegt im Intervall $[0, 1]$. Dazu werden die Layer nach ihrer Berechnung normalisiert. Abbildung 5 zeigt die Features für ein Beispielbild.

3.1.1 Grauwerte

Das Grauwert-Feature ist das einfachste Feature. Es berechnet einen Layer, in dem die Grauwerte des Bilds gespeichert werden.

3.1.2 Farbwerte

In der Klasse `ColorFeature` werden aus der Bilddatei die drei Farbkanäle auf drei Layer verteilt. Je nachdem, wie die Bilddatei aufgebaut ist, können die Farbwerte im RGB- oder im HSV-Farbraum liegen. Es wird keine Farbraum-Transformation durchgeführt. Die Farbwerte der Features in Abbildung 4 und 5 sind aus dem HSV-Farbraum.

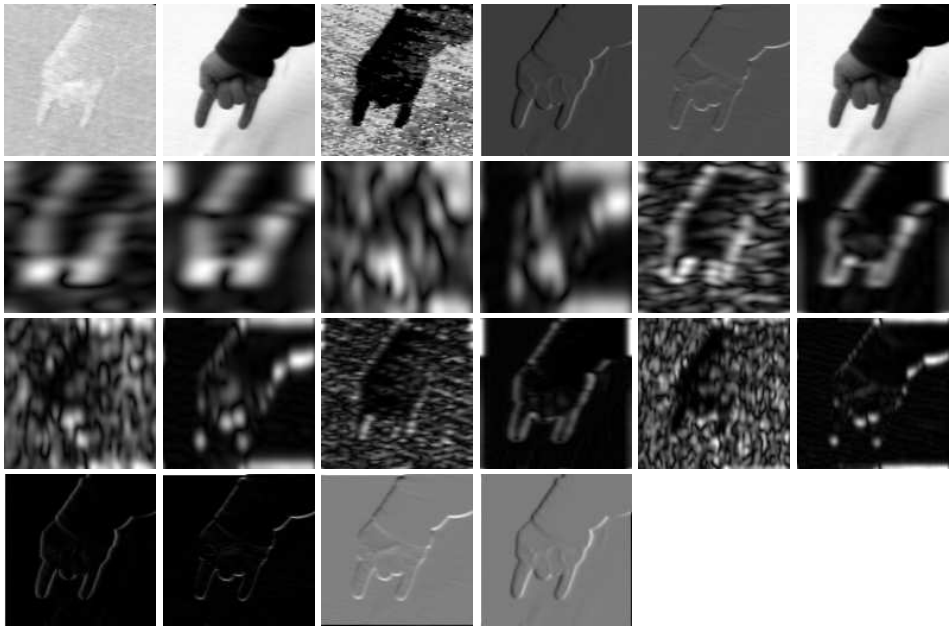

Abbildung 5: Features aus einem Bild: Farbe mit Komponenten S, V, H (3), Ableitung (2), Grauwert (1), Gabor-Transformation mit 3 Frequenzen und 2 Orientierungen jeweils für Farbe und Helligkeit getrennt (12), absolutwertige Ableitung (2), Sobel-Filter (2)

3.1.3 Ableitung

Die Klasse `DerivationFeature` berechnet die horizontale und vertikale Ableitung der Grauwerte des Bildes und speichert diese in zwei Layern. Für ein Bild $\mathbf{X}$ ist das Ableitungs-Feature eine Funktion

$$f_{Ableitung}(\mathbf{X})[x,y] = \begin{pmatrix} \mathbf{X}[x,y] - \mathbf{X}[x-1,y] \\ \mathbf{X}[x,y] - \mathbf{X}[x,y-1] \end{pmatrix} \tag{9}$$

Das Ableitungs-Feature findet „Kanten“, also Grauwert-Übergänge, im Bild. Je größer die Differenz zwischen den Grauwerten ist, desto größer ist der Wert der Ableitung.

3.1.4 Absolutwertige Ableitung

Bei der Ableitung der Grauwerte können negative Werte enstehen. Diese werden zwar durch die Normalisierung wieder auf Werte zwischen 0 und 1 abgebildet, aber bezeichnen trotzdem noch einen „negativen Kantenübergang“. Um die Auswirkung der negativen Werte auf die Klassifikation zu untersuchen, wurde die Klasse `AbsDerivationFeature` implementiert. Statt der Differenzen in Formel 9 wird der Betrag der Differenz verwendet.

3.1.5 Sobel-Filter

Der Sobel-Filter kombiniert die Differentiation mit einem Gauß-Filter. Dazu werden Masken S_h und S_v für die horizontale und vertikale Ableitung mit dem Bild gefaltet.

$$S_h = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad S_v = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \tag{10}$$

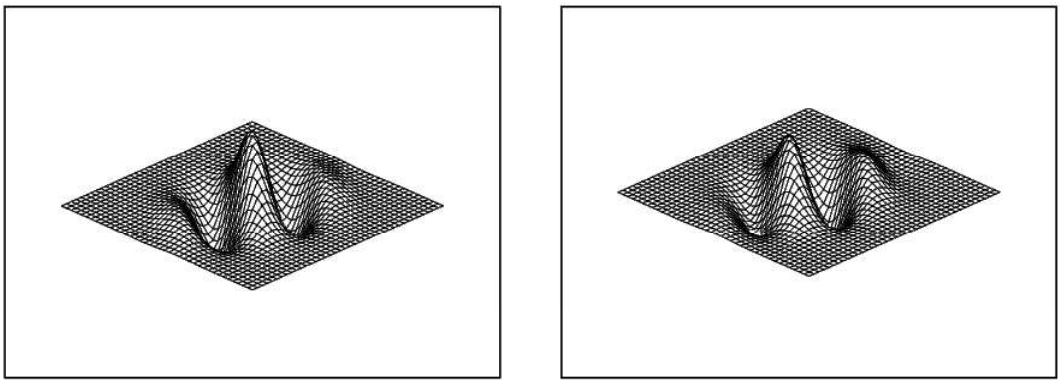

Abbildung 6: Ein Gaborfilter. Rechts der reelle Anteil, links der imaginäre. (Aus [4])

$$s(S,\mathbf{X})[x,y] = \sum_{-1\leq x'\leq 1} \sum_{-1\leq y'\leq 1} S[x'+1,y'+1]\cdot X[x+x',y+y'] \tag{11}$$

$$f_{Sobel}(\mathbf{X})[x,y] = \begin{pmatrix} s(S_h,\mathbf{X})[x,y] \\ s(S_v,\mathbf{X})[x,y] \end{pmatrix} \tag{12}$$

Die Klasse `SobelFeature` berechnet die Anwendung des Sobel-Filters und speichert das Ergebnis in zwei Layern.

3.1.6 Gabor-Transformation

Die Gabor-Transformation ist eine spezielle Fenster-Fourier-Transformation. Als Fensterfunktion wird die Gauß-Funktion

$$g_\alpha(t) = \frac{1}{2\sqrt{\pi\alpha}} e^{-\frac{t^2}{4\alpha}} \tag{13}$$

mit verschiedenen α verwendet. Das Ergebnis G_f der Gabortransformation eines Signals f, kann als Faltung mit Phasenverschiebung dargestellt werden:

$$G_f(\omega,\tau) = e^{i\omega\tau}\,(f(\tau) * h(\tau)) \tag{14}$$

Ein zweidimensionaler Gaborfilter zur Faltung, wie er für die Filterung von Bildern benötigt wird, ergibt sich aus dem Produkt zweier eindimensionaler Gaborfilter:

$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} e^{2\pi i(u_0 x + v_0 y)} \tag{15}$$

σ bezeichnet hier die Standardabweichung und $(u_0,v_0) = (f\cos(\theta), f\sin(\theta))$ mit Frequenz f und Orientierung θ ist der Frequenzmittelpunkt des Filters im Frequenzraum.

Für die Berechnung der Gabortransformationen von Farbbildern werden diese in den HSV-Farbraum mit Kanälen für Farbton (Hue), Sättigung (Saturation) und Helligkeit (Value) transformiert. Die Buntheit eines Pixels kann dann als komplexer Wert betrachtet werden:

$$b(x,y) = s(x,y)e^{i\cdot h(x,y)} \tag{16}$$

Ein Bild kann somit in ein reellwertiges Helligkeitsbild $v(x,y)$ und ein komplexwertiges Buntheitsbild $b(x,y)$ zerlegt werden. Die Gabor-Transformation wird dann sowohl auf das Helligkeits- als auch auf das Buntheitsbild angewendet. Zur Berechnung der Gabor-Transformation wird das Verfahren aus [4] angewendet. Bei diesem Verfahren wird das Eingabebild zunächst mit einer FFT in

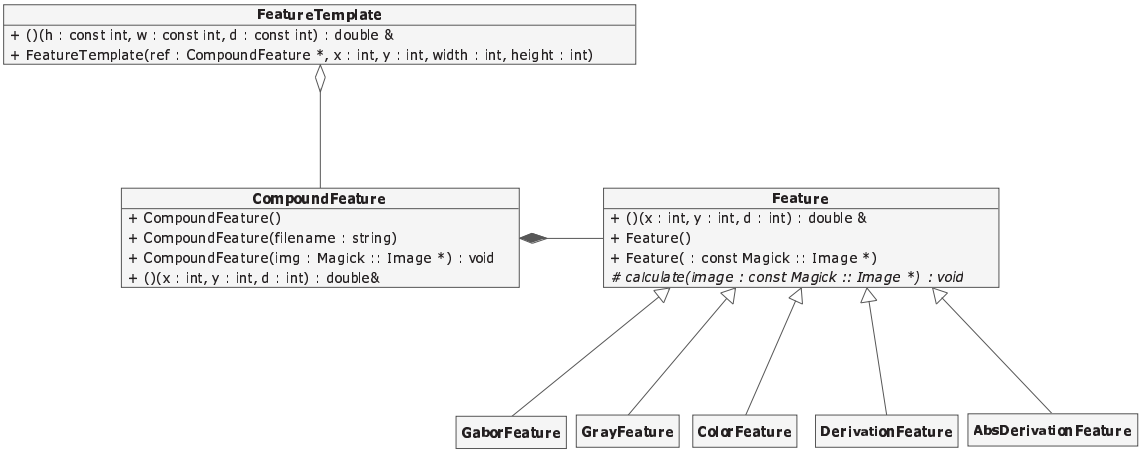

Abbildung 7: Klassendiagramm mit den Klassen für den Zugriff auf die Daten der Features. Es sind nur die wichtigsten Methoden dargestellt.

der Frequenraum transformiert. Im Frequenzraum wird das Bild dann mit dem vorher berechneten Filter multipliziert und anschließend mit inverser FFT in den Ortsraum zurück transformiert.

Im gefilterten Bild enthalten die Pixel Infomationen darüber, wie stark die gegebene Frequenz mit der gegebenen Orientierung in der Umgebung des Pixels im ursprünglichen Bild vertreten ist.

Die Klasse `GaborFeature` berechnet für eine feste Anzahl von Frequenzen und Orientierungen die Gabor-Transformationen des Helligkeits- und Buntheitsbilds. Für jede Kombination von Frequenz und Orientierung werden zwei Layer für Helligkeit- und Buntheit mit den Beträgen der komplexwertigen Ergebnisse der Gabor-Transformation gefüllt. Für n Frequenzen und m Orientierungen werden insgesamt also $2 \cdot n \cdot m$ Layer berechnet. Für Graubilder kann die Verwendung der Buntheitsbilder unterbunden werden, so dass dann nur $n \cdot m$ Layer entstehen.

Da in den Bildern aus der Bochum-Gestures Bilddatenbank (siehe Abschnitt 4.1.1) der Sättigungs-Kanal Störungen aufweist, wurde eine spezielle Klasse `BochumGaborFeature` implementiert. Diese berücksichtigt für das Buntheitsbild nur den Farbton.

3.2 Aufbau der Komponenten

Die Implementierung wurde so gewählt, dass das System möglichst flexibel ist. Neue Features können einfach in das System integriert werden. Dafür war es notwendig, den Zugriff auf die Daten der Features transparent zu gestalten. Dazu wurde die Klasse `CompoundFeature` entwickelt, die die vorhandenen Features zusammenfasst und den Zugriff auf einzelne Layer gestattet. Die Klasse `FeatureMatching`, die das Matching durchführt, greift nur auf die Klasse `CompoundFeature` zu. Alle Features sind von der Klasse `Feature` abgeleitet, die das Interface für die Features definiert und allgemeine Methoden für alle Features zur Verfügung stellt. Abbildung 7 gibt einen Überblick über die Klassen und deren Beziehungen.

Um eine Featuresammlung für ein Bild zu berechnen, wird eine neue Instanz der Klasse `CompoundFeature` erstellt. Über den Konstruktor erhält diese eine Referenz auf ein Bild im Speicher. Nun werden vom `CompoundFeature`-Objekt Instanzen der Feature-Klassen erzeugt, denen die Referenz auf das Bild mitgeteilt wird. Beim Erzeugen der Features wird deren Methode `calculate(Image *image)` aufgerufen. Diese Methode, die die Berechnung des Features durchführt, ist in der Basisklasse `Feature` abstrakt definiert und muss von jeder abgeleiteten Klasse überschrieben werden. Die Feature-Objekte teilen dem CompoundFeature mit, wie viele Layer sie zur Verfügung stellen. Daraus kann das CompoundFeature eine Abbildung der insgesamt zur

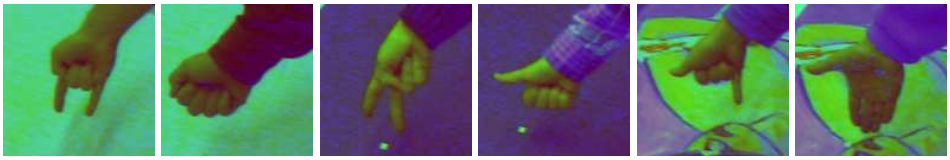

Abbildung 8: Bilder aus der Bochum-Gestures-Datenbank

Verfügung stehenden Layer auf die einzelnen Feature-Objekte berechnen.

Sind alle Features erzeugt, kann das `CompoundFeature`-Objekt in eine Datei gespeichert werden. Da das Objekt, abhängig von der Anzahl der Features und der Größe des Bildes, relativ viele Daten beinhaltet, wurde ein komprimiertes Dateiformat (Deflation-Methode aus der Bibliothek zlib) gewählt.

Der Zugriff auf die Daten einer Featuresammlung erfolgt über den `()`-Operator `(int x, int y, int d)` der Klasse `CompoundFeature`. Der Zugriff auf den Layer d wird an ein Feature-Objekt delegiert, indem das Objekt bestimmt wird, das diesen Layer enthält. Die Feature-Klassen besitzen ebenfalls einen `()`-Operator `(int x, int y, int d)` mit dem auf den Wert an Position (x,y) des Layers d zugegriffen werden kann. Allerdings bezeichnet d hier einen Layer, den das Feature gefüllt hat, während in der Klasse CompoundFeature auf alle Layer zugegriffen werden kann.

Ein Template ist ein Ausschnitt eines Bildes beziehungsweise einer Featuresammlung. Dementsprechend wurde die Klasse `FeatureTemplate` gestaltet, die ein Template modelliert. Sie enthält lediglich eine Referenz auf ein `CompoundFeature`-Objekt und die Koordinaten an denen das Template auf dem Bild positioniert ist. Die Art des Datenzugriffs eines Templates ist gleich mit dem eines CompoundFeatures; es werden lediglich die Koordinaten verschoben.

4 Ergebnisse

4.1 Bilddatenbanken

Die Experimente die in den folgenden Abschnitten beschrieben werden, wurden auf drei Bilddatenbanken durchgeführt:

4.1.1 Bochum-Gestures

Die Bochum-Gestures Bilddatenbank besteht aus Fotos von Handgesten. Insgesamt sind 1036 Farbbilder mit 128×128 Pixeln enthalten. Die Bilder sind aufgeteilt in 12 Klassen (12 Handgesten). Jede Geste liegt von unterschiedlichen Personen jeweils vor hellem, dunklem und komplexem Hintergrund vor. Die Bilddatenbank wird in [5] vorgestellt. Beispielbilder zeigt Abbildung 8.

Die Autoren von [5] verwenden Elastic Graph Matching zur Erkennung der Handgesten. Dabei wird aus den Trainingsbildern ein so genannter Compound Bunch Graph berechnet, dessen Knoten ausgezeichnete Regionen des Objekts beschreiben. Die Regionen, für die Knoten erstellt werden, sind manuell markiert. Jeder Knoten ist mit einem Jet beschriftet, der aus Anworten von Gabor-Filtern und Farb-Merkmalen besteht. Die im Training berechneten Graphen können dann mit den Graphen der zu klassifizierenden Bilder verglichen werden. Mit diesem Verfahren werden Fehlerraten von 7,1% für Bilder mit einfachem Hintergrund und 14,2% für Bilder mit komplexem Hintergrund erzielt. Diese Ergebnisse sind nicht mit den in dieser Arbeit durchgeführten Experimenten zu vergleichen, da die Bilder dort manuell für das Training bearbeitet wurden.

Abbildung 9: Bilder aus der CALTECH-Faces-Datenbank

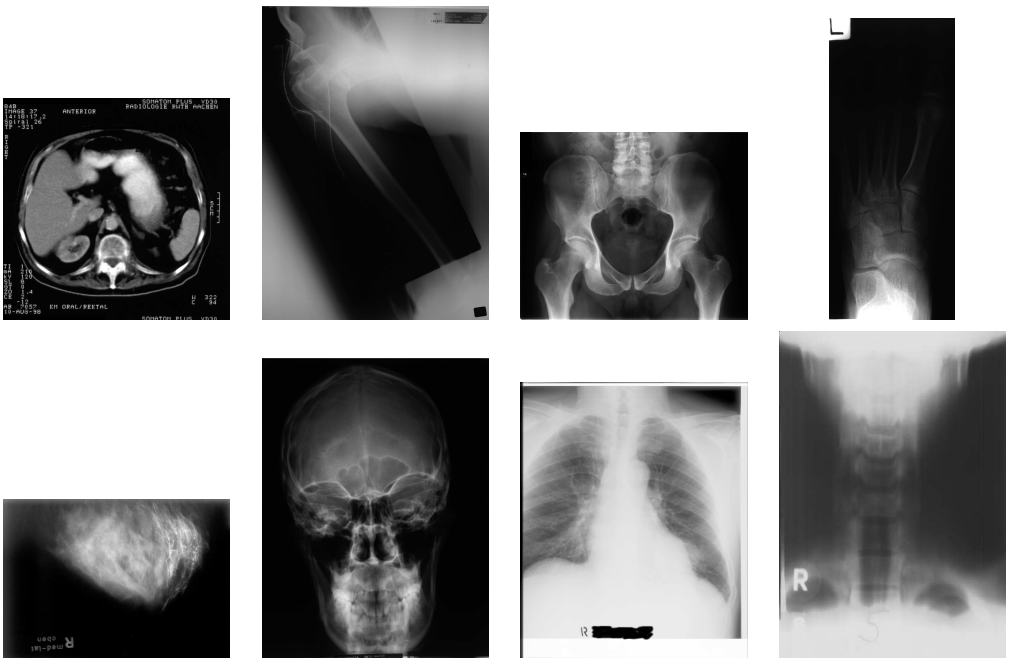

Abbildung 10: Bilder aus der IRMA-Datenbank

4.1.2 CALTECH-Faces

Das California Institute of Technology hat eine Bilddatenbank erstellt, in der Fotos von Gesichtern vor komplexen und vor einfachen Hintergründen gespeichert sind [6]. Für die Experimente wurden 534 dieser Fotos verwendet. In Abbildung 9 sind Bilder aus der Datenbank dargestellt.

4.1.3 IRMA

Die IRMA-Datenbank enthält Röntgenaufnahmen von menschlichen Körperteilen. Die Datenbank liegt in verschiedenen Größen und mit unterschiedlich vielen Klassen vor. Für die Experimente in dieser Arbeit wurde die Datenbank mit 3879 Bildern verwendet, die in 26 Klassen unterteilt sind. Bilder einer Klasse zeigen das gleiche Körperteile sind aber aus unterschiedlichen Perspektiven aufgenommen und haben unterschiedliche Größen (siehe Abbildung 10). Außerdem sind für jede Klasse verschieden viele Trainingsbilder vorhanden. Eine detaillierte Beschreibung der Datenbank findet sich in [7].

Tabelle 1: Fehlerraten für Experimente auf der Bochum-Gestures Datenbank. Angegeben sind die mittleren Fehlerraten der drei Testdatensätze.

Features	Fehlerrate [%]
Gray	8,6
Gray, Derivation	12,5
Gray, Sobel	10,3
Color	21,4
Color, Derivation	21,7
Gray, Gabor (3 Frequenzen, 2 Orientierungen, H- und V-Kanal)	4,7

Tabelle 2: Fehlerraten auf der Bochum-Gestures-Datenbank in verschiedenen Testdatensätzen.

Features	Testset	Fehlerrate [%]
Color	1	26,7
Color	2	9,2
Color	3	28,2
Color, Derivation	1	30,0
Color, Derivation	2	6,7
Color, Derivation	3	28,3

4.2 Experimente

Ziel der Experimente war es, die Auswirkung der Benutzung mehrerer Features auf die Fehlerraten zu beobachten. In einem Vorverarbeitungsschritt wurden für alle Bilder CompoundFeatures mit unterschiedlichen Features erstellt.

Für die Experimente auf der Bochum-Gestures-Datenbank wurden die Bilder mit hellem Hintergrund verwendet. Die 345 Bilder mit 12 Gesten wurden in 3 Testdatensätze aufgeteilt. In jedem Testset wurden 120 Bilder (10 pro Klasse) für den Test verwendet und der Rest für das Training. Für die Startmodelle wurde aus allen Trainingsbildern einer Klasse der Mittelwert berechnet und daraus CompoundFeatures. Das Hintergrundmodell wurde als Gleichverteilung mit geringem Gewicht in die Distanzberechnung einbezogen. Mit einem Mean-Variance-Treshold von 0.1 wurden Dichten vermieden, die visuell gleichförmig sind. Zur Beschleunigung wurden die Bilder im ersten Schritt des Matchings auf 60×60 Pixel skaliert. In Tabelle 1 sind ausgewählte Ergebnisse der Experimente dargestellt. Zur besseren Übersicht sind nur die mittleren Fehlerraten aus den jeweils 3 Testdatensätzen dargestellt. Tabelle 2 zeigt für zwei Feature-Kombinationen die Fehlerraten in unterschiedlichen Testdatensätzen. Die Ergebnisse der Experimente mit dem Gabor-Feature und unterschiedlich vielen Orientierungen sind in Tabelle 3 aufgeführt.

Da die CALTECH-Faces-Datenbank keine Klasseninformationen beinhaltet, wurden nur Modelle für eine Klasse „Gesicht“ trainiert. Im Test wurden Bilder mit und ohne Gesicht klassifiziert. Anschließend wurde ein Grenzwert für die ermittelten Distanzen berechnet, mit dem entschieden werden kann, ob ein Bild ein Gesicht enthält oder nicht. Liegt die Distanz unter dem Grenzwert, wird das Bild der Klasse „Gesicht“ zugewiesen, sonst der Klasse „Kein Gesicht“. Der beste Grenzwert wurde automatisch ermittelt, indem für eine Menge von Grenzwerten die resultierenden Fehlerraten berechnet wurden. Da es sich um nicht um Farbbilder handelt, wurden im Gabor-Feature

Tabelle 3: Fehlerraten auf der Bochum-Gestures-Datenbank mit den Features Gray und Gabor.

Frequenzen	Orientierungen	Fehlerrate [%]
3	2	4,7
3	4	5,0
3	5	5,8

Tabelle 4: Fehlerraten auf der CALTECH-Faces-Datenbank. Die Gewichtung ist pro Layer angegeben.

Features	Gewichtung	Fehlerrate [%]
Gray	1	47,7
Gray, Abs. Derivation	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	42,2
Gray, Abs. Derivation	$\frac{1}{2}, \frac{1}{4}, \frac{1}{4}$	42,6
Gray, Derivation	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	32,3
Gray, Derivation	$\frac{1}{2}, \frac{1}{4}, \frac{1}{4}$	33,6
Gray, Gabor (2 Freq., 1 Orient.)	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	42,2

die Layer für die Buntheits-Bilder deaktiviert. Die 179×118 Pixel großen Bilder wurden in der ersten Stufe des Matchings um 30% verkleinert. Auch hier wurde das Hintergrundmodell gering gewichtet.

Um die Bilder der IRMA-Datenbank zu benutzen, wurde diese auf eine feste Breite von 32 Pixeln skaliert. Das Seitenverhältnis blieb erhalten. Es wurden Modelle für alle 26 Klassen ohne initiale Modelle trainiert. Für den Test wurden 1016 Bilder benutzt. Für das Matching wurden die Bilder nicht skaliert. Das Hintergrundmodell wurde wieder gering bewertet.

4.3 Analyse der Ergebnisse

Aus den Fehlerraten in Tabelle 1 ist ersichtlich, dass die Informationen des Gabor-Features die Erkennung verbessern. Alle anderen untersuchten Kombinationen von Features liefern Fehlerraten, die größer sind, als die des Experiments mit Grauwerten. Abbildung 13 zeigt die steigende Fehlerrate bei wachsendem Einfluss des Sobel-Features. Die erwartete Verbesserung der Erkennungsleistung bei steigender Anzahl von Features bestätigt sich also, tritt aber nicht in allen Experimenten auf. In Tabelle 3 sieht man, dass sich das Fehlerrate durch mehr Layer im Gabor-Feature nicht weiter senken läßt.

Die Werte in Tabelle 2 zeigen, dass die Fehlerraten stark schwanken, je nachdem welche Bilder für das Training bzw. für den Test verwendet werden. Die Modelle repräsentieren relativ wenige Varianten der Gesten, weil pro Klasse nur 18 bis 19 Bilder für das Training zur Verfügung stehen. Zudem sieht man in Abbildung 11, dass die Gesten zum Teil sehr ähnlich sind, was die Erkennung erschwert. Die Abbildung zeigt jedoch auch, dass es dem Verfahren gelingt, den relevanten Bildausschnitt vollautomatisch zu bestimmen.

In den Experimenten auf den Gesichts-Bildern senkt die Verwendung der Ableitung die Fehlerrate. Die Verwendung des Gabor-Features lässt die Fehlerrate wieder ansteigen, bleibt aller-

Tabelle 5: Fehlerraten auf der IRMA-Datenbank.

Features	Fehlerrate [%]
Gray	59,6
Gray, Derivation	58,6
Gray, Derivation, Gabor (2 Frequ., 1 Orient.)	63,4

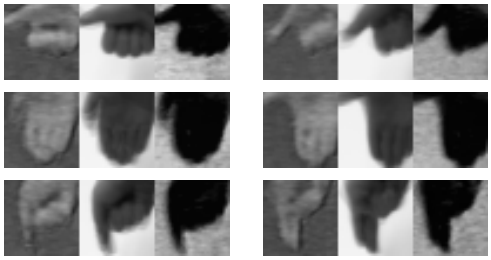

Abbildung 11: Modelle für die Klassen 2, 12, 3, 4, 5 und 7 der Bochum-Gestures. Trainiert mit dem Feature Color.

dings unter der Fehlerrate des Experiments, in dem nur Grauwerte benutzt werden. In Tabelle 4 sieht man, dass die Verwendung unterschiedlicher Gewichtungen für die Features keine deutliche Änderung der Fehlerrate bewirkt. Mit speziellen Verfahren zur Gesichtserkennung können deutlich bessere Fehlerraten erzielt werden. Der hier vorgestellte Ansatz macht jedoch keinerlei Annahmen zur Struktur der erkannten Objekte.

Bei den Ergebnissen für die Experimente auf der IRMA-Datenbank in Tabelle 5 fällt auf, dass die Fehlerraten durchgängig sehr hoch sind. Das liegt daran, dass in den einzelnen Klassen sehr unterschiedliche Aufnahmen vertreten sind. Allerdings reicht die Anzahl der Bilder pro Art der Aufnahme in der Regel nicht, um eine eigene Dichte zu trainieren. Für manche Klassen sind zudem nur sehr wenige Trainingsbilder vorhanden. In Abbildung 12 sieht man, dass die trainierten Modelle visuell nicht aussagekräftig sind. Mit diesem Modellen können die Bilder offensichtlich nicht richtig klassifiziert werden.

5 Fazit und Ausblick

Das vorgestellte System ermöglicht die Integration mehrerer Features in einen holistischen Bilderkenner. Durch die Verwendung mehrerer Features läßt sich die Erkennungsleistung verbessern.

Offenbar ist der Einfluss der Features auf die Fehlerrate abhängig von der Art der untersuchten

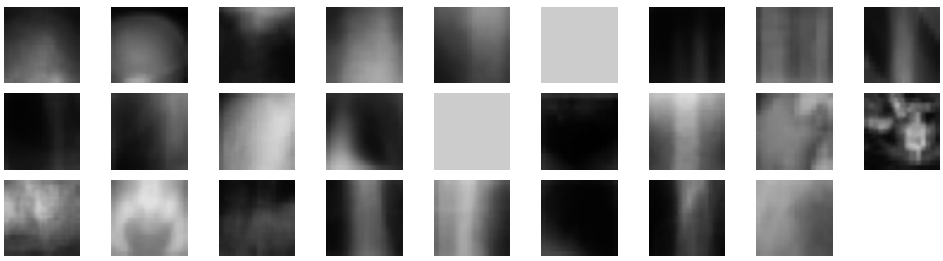

Abbildung 12: Modelle für die 26 Klassen der IRMA-Datenbank

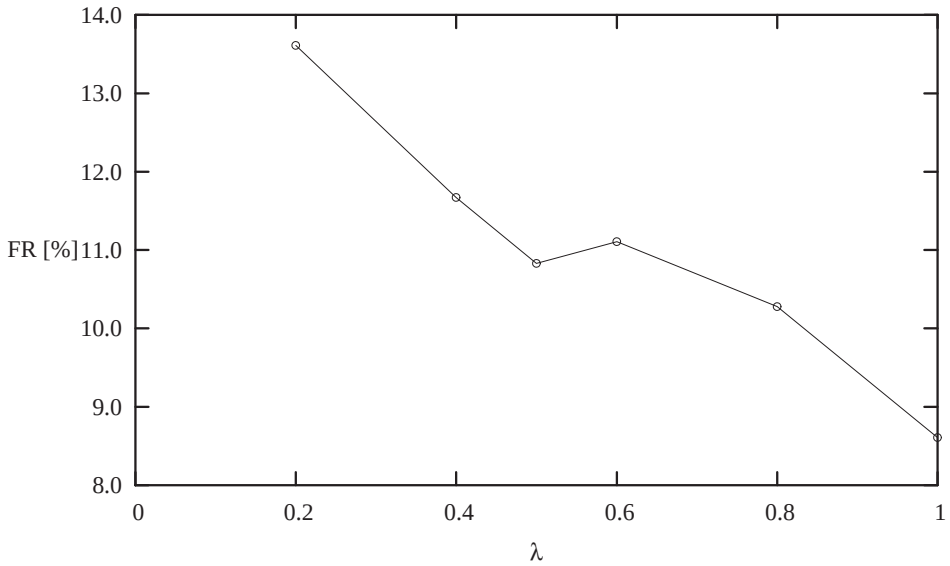

Abbildung 13: Fehlerraten (FR) auf der Bochum-Gestures-Datenbank mit den Features Gray und Sobel und Gewichten λ für den Gray- und $\frac{1-\lambda}{2}$ für die Sobel-Layer.

Bilder bzw. der Bilddatenbank. In der Bochum-Gestures Datenbank sind einige Gesten untereinander sehr ähnlich und somit schwer zu differenzieren. Trotzdem erzielt das System mit einer Kombination aus Grauwert- und Gabor-Features niedrige Fehlerraten. Die relativ kleinen Trainingsdaten könnten vergrößert werden, indem Variationen durch geringe Rotation und Skalierung erstellt werden. In der Datenbank liegen Bilder mit drei Hintergrundtypen vor. Zum Training und zum Testen könnten Bilder mit unterschiedlichem Hintergrund gemischt werden. Mit dem vorgestellten Verfahren würde sich die Fehlerrate dadurch aber verschlechtern, weil das Objekt nur grob segmentiert werden kann und der Hintergrund dadurch zum Teil in die Modelle einfließt. Dadurch würden für jeden Hintergrundtyp einzelne Dichten erzeugt, die jeweils wieder mit relativ wenig Daten trainiert sind.

Auch in der IRMA-Datenbank liegen für viele Klassen nicht genug Trainingsdaten vor. Zudem sind die Klassen für dieses Verfahren zu allgemein gehalten. Die Variationen von Aufnahmeposition und Bildausschnitt können nicht oder nur schlecht modelliert werden. Auch ist die Form der Objekte nicht geeignet, um sie mit quadratischen Prototypen zu erfassen.

Abhilfe für das Problem der schlechten Segmentierung beziehungsweise der Abhängigkeit vom Hintergrund, könnten nicht-quadratische Prototypen schaffen. Dazu könnten zunächst quadratische Prototypen berechnet und in einem weiteren Schritt verfeinert werden. In [8] wird ein Verfahren beschrieben, das Objekte auf heterogenem Hintergrund mit nicht-quadratischen Prototypen erkennt.

Bislang berücksichtigt das Verfahren noch keine Rotation der zu erkennenden Objekte. Bereits bei der Berechnung der Feature-Sammlungen, also im Vorverarbeitungsschritt, könnten für die Trainingsdaten Features aus rotierten Bildern berechnet werden. Eine Rotation der Features ist nicht direkt möglich, da einige Features von der Ausrichtung der Originaldaten abhängen.

Durch eine Glättung der Bilder kann das „Rauschen“ entfernt oder vermindert werden. Dies ist durch den Einsatz eines Gauß-Filters vor der Berechnung der Features möglich. In anderen Experimenten konnte dies die Fehlerrate senken.

Die Erkennungsraten könnten, abgesehen von den oben genannten Ansätzen, auch durch eine

andere Wahl der Parameter verbessert werden. Ebenso mag eine andere Wahl der Gewichte die Fehlerraten senken. Ziel dieser Arbeit war es aber nicht, möglichst gute Fehlerraten zu erzielen, sondern die Integration unterschiedlicher Features in die bestehende Software zu untersuchen. Eine Feinabstimmung der Parameter würde den Rahmen dieser Arbeit sprengen.

Auch bei der Implementierung gibt es noch Verbesserungsmöglichkeiten, die die Handhabung vereinfachen. Derzeit ist es nicht möglich, die Zusammenstellung der Features zur Laufzeit zu bestimmen. Um dies zu ändern, müssten die von der Klasse CompoundFeature verwendeten Feature-Klassen beim Start des Programms initialisiert werden. Die Daten der Features sollten dann in separaten Dateien gespeichert und bei Bedarf geladen werden können.

A Dokumentation der Software

A.1 Merkmalsberechnung (`calcfeatures`)

`calcfeatures` wird in der Vorverarbeitung zum Berechnen der Features verwendet.
Aufruf: `calcfeatures filelist rotationlist directory new-filelist`
`filelist` ist eine Datei in der alle Bilddateien verzeichnet sind, für die Features berechnet werden sollen. Sie hat folgende Aufbau:

```
Anzahl-Dateien
Klassen-Nr. Dateiname_Bild_1
Klassen-Nr. Dateiname_Bild_2
...
```

Mit `rotationlist` kann eine Datei angegeben werden, in der Rotationswinkel angegeben sind. Das Format ist ähnlich zu dem für `filelist`, nur werden keine Klassen-Nummern angegeben. Wenn Rotationen angegeben sind, wird für jedes Bild in jeder angegebenen Rotation ein Feature berechnet. Alternativ kann `0` übergeben werden, wenn keine Rotation durchgeführt werden soll. Die Feature-Dateien werden im Verzeichnis `directory` abgelegt. `new-filelist` ist die Datei, in die eine Liste mit den neuen Feature-Dateien geschrieben wird. Die Klassennummern werden übernommen, die Dateipfade werden an `directory` angepasst.

A.2 Training (`train`)

Mit dem Programm `train` wird das in Abschnitt 2.4 beschriebenen Training durchgeführt. Aus einer Menge von Feature-Dateien wird eine Gauß'sche Mischverteilung für das Objekt und eine Gaußverteilung für den Hintergrund berechnet. Die Optionen sind in Tabelle 6 aufgeführt.

A.3 Klassifikator (`class`)

Das Programm `class` führt die Objektklassifikation für mehrere Bilder durch und bestimmt die Fehlerrate. Die Optionen sind in Tabelle 7 angegeben.

A.4 Merkmalsvisualisierug (`showfeature`)

Mit `showfeature` kann aus einer Feature-Datei eine Bilddatei erzeugt werden. Das Programm wird mit

```
showfeature feature-file prefix-for-output [-one]
```

aufgerufen. `feature-file` ist der Dateiname der Feature-Datei. Es werden Bilddateien (im PNG-Format) erstellt, die als Dateinamen das Präfix `prefix-for-output` mit angehängtem Namen des

Features haben. Wird die Option `-one` angehängt, wird eine Bilddatei erzeugt, in der alle Feature-Bilder nebeneinander gezeigt werden. Das `prefix-for-output` bezeichnet dann den ganzen Dateinamen.

A.5 Konfiguration

Die Konfiguration der Features erfolgt vor der Kompilation der Software. Alle kompilierten Programme sind dann für die konfigurierte Zusammenstellung der Features angepasst. Zur Konfiguration existiert das Skript `configure`, das die in Tabelle 8 aufgelisteten Optionen hat. Features werden mit `--enable-feature-<name>` aktiviert und mit `disable-feature-<name>` deaktiviert. Folgende Features stehen zur Verfügung: `gray`, `color`, `derivation`, `absderivation`, `gabor`, `sobel`, `bochumgabor`. Neben den gezeigten Optionen stehen Standard-Optionen für Installationspfade, Compileroptionen etc. zur Verfügung, welche in der Hilfe beschrieben sind, die mit `configure --help` angezeigt werden kann.

Literatur

[1] M. Motter, “Statistische Modellierung von Bildinhalten für die Bilderkennung,” Diplomarbeit, Lehrstuhl für Informatik VI, RWTH Aachen, Dezember 2001.

[2] H. Ney, “Pattern recognition and neural networks,” Lecture notes, RWTH Aachen, Lehrstuhl für Informatik VI, 2003.

[3] D. Keysers, M. Motter, T. Deselaers, and H. Ney, “Training and recognition of complex scenes using a holistic statistical model,” in *Proceedings of the 25th DAGM-Symposium Pattern Recognition*, Lecture Notes in Computer Science 2781, pp. 52–59, Springer Verlag, September 2003.

[4] D. Keysers, “Textureanalyse von Farbbildern mit Gaborfiltern,” Studienarbeit, Institut für medizinische Informatik, RWTH Aachen, Juni 1999.

[5] J. Triesch and C. von der Malsburg, “A system for person-independent hand posture recognition against complex backgrounds,” *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 23, pp. 1449–1453, November 2001.

[6] M. Burl, T. Leung, and P. Perona, “Face localization via shape statistics,” in *Proc. Int. Workshop on Automatic Face and Gesture Recognition*, pp. 154–159, Juni 1996.

[7] T. Lehmann, M. Güld, C. Thies, B. Fischer, K. Spitzer, D. Keysers, H. Ney, M. Kohnen, H. Schubert, and B. Wein, “Content-based image retrieval in medical applications,” *Methods of Information in Medicine*, vol. 43, pp. 354–361, März 2004.

[8] M. Reinhold, D. Paulus, and H. Niemann, “Appearance-based statistical object recognition by heterogeneous background and occlusions,” in *Proceedings of the 23rd DAGM-Symposium Pattern Recognition*, pp. 254–261, Springer-Verlag, September 2001.

Tabelle 6: Optionen für `train`

Option	**Beschreibung**
`--acc <val>`	Im ersten Schritt des Matchings werden die Features auf `<val>` % der Bildgröße skaliert.
`--bgMean <val>`	Initialer Mittelwert für das Hintergrundmodell
`--bgVariance <val>`	Initiale Varianz für das Hintergrundmodell
`--fgMeanImage <file>`	Feature-Datei für einen initialen Mittelwert für das Objektmodell
`--fgVariace <val>`	Initiale Varianz für das Objektmodell
`--maxObjSize <val>`	Maximale Größe von Templates, die gesucht werden
`--minObjSize <val>`	Minimale Templategröße
`--maxHyp <val>`	Maximale Anzahl von Hypothesen, die weiterverfolgt werden
`--noIterations <val>`	Anzahl der Iterationen, die höchstens durchgeführt werden, ohne dass die Distanz konvergiert (siehe Abschnitt 2.4).
`--scaleLevels <val>`	Anzahl der Skalierungsstufen, die berücksichtigt werden sollen
`--trainCorpus <datei>`	Datei in der die Liste der Trainingsdaten gespeichert ist. Das Dateiformat ist in Abschnitt A.1 beschrieben.
`--meanVarianceTreshold <val>`	Mean-Variance-Treshold, der benutzt werden soll
`--weights <datei>`	Datei in der die Gewichte für die einzelnen Layer gespeichert sind. In der Datei muss zuerst die Anzahl der Gewichte stehen und danach für jeden Layer ein Gewicht. Wird `--weights` nicht angegeben, werde alle Layer gleich gewichtet.
`--bgWeight <val>`	Gewichtung für das Hintergrund-Modell
`--maxDensities <val>`	Maximale Anzahl von Dichten in der Verteilung des Objektmodells
`--minObservations <val>`	Minimale Anzahl der Beobachtungen pro Dichte
`--noEMIterations <val>`	Anzahl der Iterationen im EM-Algorithmus

Tabelle 7: Optionen für `class`

Option	**Beschreibung**
`--acc <val>`	Im ersten Schritt des Matchings werden die Features auf `<val>` % der Bildgröße skaliert.
`--maxObjSize <val>`	Maximale Größe von Templates, die gesucht werden
`--minObjSize <val>`	Minimale Templategröße
`--maxHyp <val>`	Maximale Anzahl von Hypothesen, die weiterverfolgt werden
`--models <datei>`	Datei, in der die Dateinamen der Modelle gespeichert sind, die im Training erstellt wurden. In der ersten Zeile ist die Anzahl der Modelle gespeichert. Danach steht in jeder Zeile die Klassen-Nummer, der Dateiname des Objektmodells und der Dateiname des Hintergrundmodells
`--scaleLevels <val>`	Anzahl der Skalierungsstufen, die berücksichtigt werden sollen
`--weights <datei>`	Datei, in der die Gewichte für die einzelnen Layer gespeichert sind. Siehe Tabelle 6
`--bgWeight <val>`	Gewichtung für das Hintergrund-Modell
`--writePositions`	Schreibt Bild-Dateien mit den besten Templates

Tabelle 8: Optionen für `configure`

Option	**Beschreibung**
`--with-gabor-frequencies=<num>`	Anzahl der Frequenzen, für die Gaborfilter erstellt werden
`--with-gabor-phases=<num>`	Anzahl der Orientierungen, für die Gaborfilter erstellt werden
`--with-gabor-use-hs=(yes|no)`	Bei `yes` verwendet das Gabor-Feature Buntheitsbilder
`--with-magick=<pfad>`	Installationsort der ImageMagick++-Bibliothek
`--with-blitz=<pfad>`	Installationsort der Blitz-Bibliothek