Formerkennung anhand hierarchisch fragmentierter Konturen durch ein Ensemble von LCC-Klassifikatoren

Diplomarbeit

vorgelegt von

Alexandra Barchunova

Lehrstuhl für Kognitive Systeme
Institut für Informatik
der Christian-Albrechts-Universität zu Kiel

Betreuer:
Dipl.-Ing. Herward Prehn
Prof. Dr. Gerald Sommer

Kiel, August 2007

Inhaltsverzeichnis

1 Einführung

Im Bereich der Objekterkennung und -klassifikation wurde in den letzten dreißig Jahren umfangreich geforscht. Eines der großen Ziele ist ihre Automatisierung. Die Problemstellungen, Methoden und Objekte unterscheiden sich deutlich. Klassifiziert werden u.a. Insekten [33], arabische Schriftzeichen [13], Haushaltswerkzeuge [24], Meeresbewohner [5] oder auch Gehirne [29]. Ein großes Interesse besteht an einer schnellen Erkennung gelenkiger Objekte, z.B. Panzer [37]. Die Anwendungen finden sich hauptsächlich in der Industrie, Medizin und im Militär.

Diese Arbeit wurde im Rahmen des EU Forschungsprojekts COSPAL durchgeführt und beschäftigt sich mit der kontur-basierten Kassifikation teilweise verdeckter dreidimensionaler Figuren. Zur Hardware-Ausstattung der Experimente gehört ein Roboterarm mit einer am Ausleger montierten Kamera. Diese beobachtet die zu klassifizierenden Objekte aus verschiedenen Blickwinkeln. Zu Testzwecken werden einfache geometrische Figuren verwendet: Prismen, Zylinder, Quader, Halbzylinder und Brücken (siehe Abbildung 1).

Partielle Verdeckung ist eine große Herausforderung für die kontur-basierte Objekterkennung. Durch eine einfache Verdeckung wird das Objekt in einer lokalen Umgebung nicht sichtbar. Betrachtet man seine Kontur, so wird das entsprechende Stück durch die Form der Verdeckung ersetzt. Ein Mensch kann eine solche Kontur nur dann richtig zuordnen, wenn er lokal die Struktur betrachtet und ein Konturstück finden kann, das eindeutig einer der Figurklassen zugeordnet werden kann. Analog erfordert die Klassifikation von Konturen teilweise verdeckter Objekte innerhalb dieser Arbeit eine Methode, die eine lokale Strukturanalyse ermöglicht. Ein Verfahren, das die Konturdaten für die Extraktion verschiedener lokaler Merkmale vorbereitet, ist die **hierarchische Fragmentierung**. Aus einer Kontur erzeugt dieses Verfahren mehrere Fragmente-Ebenen, wobei die strukturelle Komplexität der Konturstücke von Ebene zur Ebene ansteigt.

Um einen lokalen Ansatz bei der Erkennung teilweise verdeckter Konturen zu ermöglichen, wird innerhalb dieser Arbeit ein Klassifikator-Ensemble **Adaptive Occlusion Classifier** (AOC) entworfen und implementiert. Sowohl beim Training als auch im Testvorgang dient das Ensemble, aufbauend auf dem Verfahren zur hierarchischen Fragmentierung, als eine Grundlage für die lokale Strukturanalyse. Die Mitglieder des Ensembles, eine Reihe der sog. LCC-Klassifikatoren, sind spezialisiert auf eigene Verdeckungsstufen, die durch

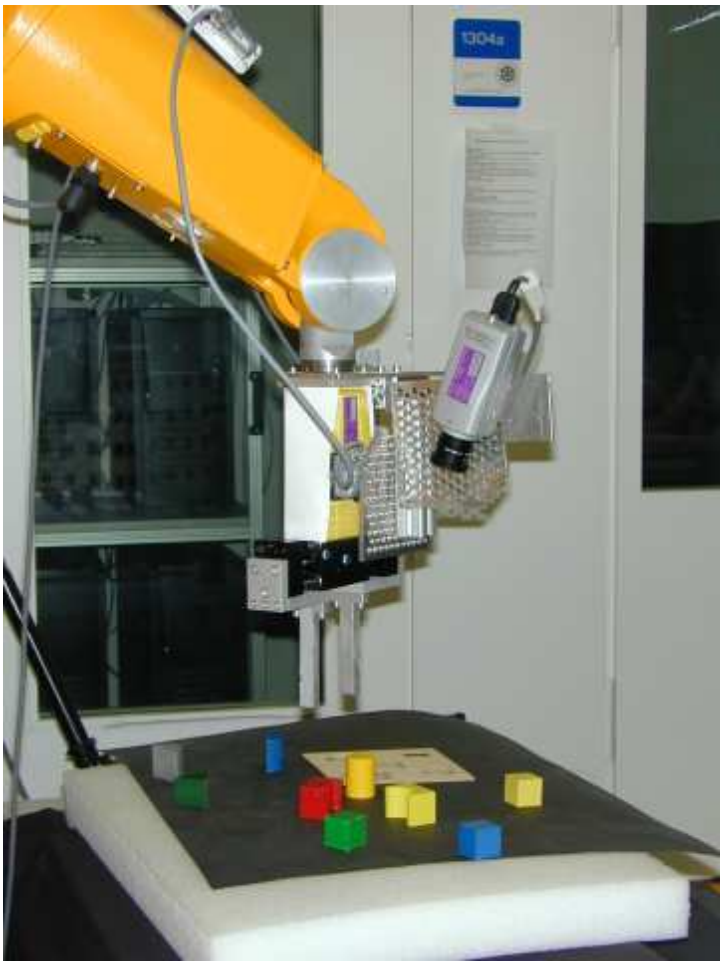


Abbildung 1: Experimentelles Setup bei COSPAL

die Ebenen der hierarchischen Fragmentierung definiert werden. Das Ensemble verwendet die *class-consious weighted average* Methode, um die Endhypothese aus den Antworten der Mitglieder zu ermitteln. Dafür wird mit Hilfe eines linearen Optimierungsverfahrens und anhand einer Menge verdeckter Konturen ein Gewichtsvektor ermittelt, der für die Gewichtung der Klassenantworten verwendet wird. Dies hat sich empirisch (siehe Kapitel 6) als erfolgreich erwiesen. Aus dem Einsatz des AOC Ensembles anstelle eines einzelnen LCC-Klassifikators resultiert eine deutliche Verbesserung der Klassifikationsergebnisse.

Für die Kontur-Repräsentation im Klassifikationsvorgang benötigt man eine Methode zur Merkmalsextraktion. Die hierfür verwendeten Fourier-Deskriptoren, Zernike- und pseudo Zernike-Momente sind unabhängig von den affinen Transformationen wie Rotati-

on, Skalierung und Translation. Konturverzerrungen, die durch die Veränderung des Kamerablickwinkels zustande kommen, müssen durch das Ensemble-Training kompensiert werden. Im Kapitel 2 sind die Invarianz-Eigenschaften der Fourier-Deskriptoren, Zernike- und pseudo Zernike-Momente dargelegt und ihre Anwendung auf geschlossene Konturen beschrieben.

Objektkonturen, die aus RGB-Bildern erhalten werden, sind wegen einiger physikalischer und algorithmischer Faktoren verrauscht. Innerhalb dieser Arbeit wird ein Verfahren zu Konturvorverarbeitung vorgeschlagen (Kapitel 3), das die Kontur-Modellierung und die Datenformatierung für die Merkmalsextraktion enthält.

Die Merkmalsextraktion und Konstruktion der Merkmalsvektoren ist im Kapitel 4 erläutert. Im Kapitel 5 findet man eine Beschreibung der theoretischen Konzepte, Annahmen, Struktur und Aufbau des Adaptive Occlusion Classifier Ensemble. Die Ergebnisse der Experimente für die entstandene Software werden in Kapitel 6 vorgestellt. Eine schematische Darstellung der einzelnen Schritte des AOC-Algorithmus ist in Abbildung 2 zu sehen.

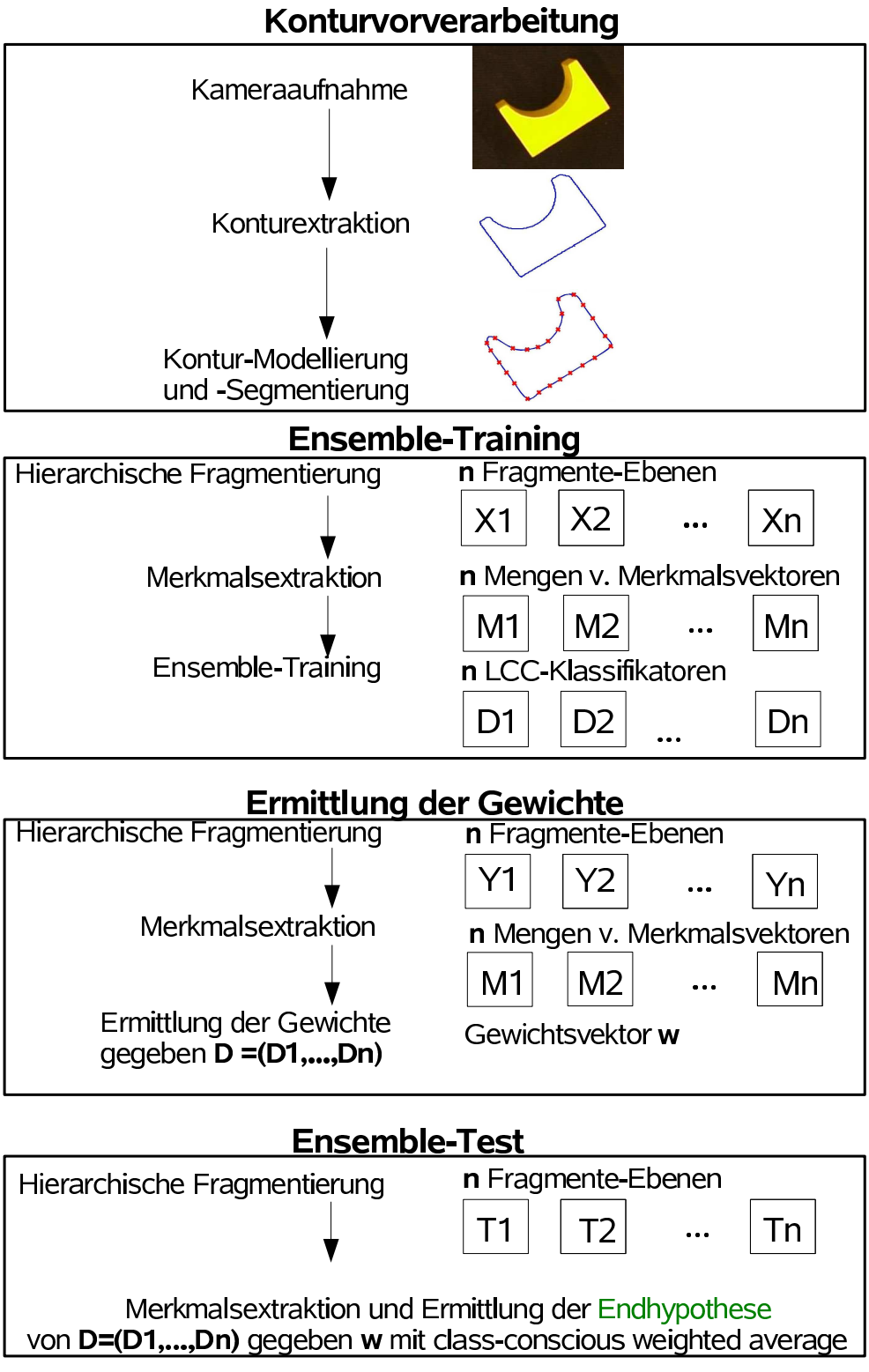


Abbildung 2: Eine schematische Darstellung des AOC-Algorithmus

2 Invarianz-Konzepte

Die Theorie der Invarianz im Bereich Computer-Vision spielt eine große Rolle seit der Gründung des Fachgebiets um 1960 [25]. Ausgiebige Forschung wurde zu diesem Thema durchgeführt. Eine Zusammenstellung der Publikationen findet man in mehreren Werken (u.a. in [26], [30], [25]), die aus den 1990er Jahren stammen. Eine der wichtigsten Aufgaben besteht darin, das Objekt trotz der Veränderung seiner Position, Größe oder Ausrichtung anhand der auf die Bildebene projizierten Daten zu erkennen. Dabei ist die Menge der Transformationen unendlich groß. Also erfordert die Automatisierung des Erkennungsprozesses eine Formalisierung der Eigenschaften, die unter den Transformationen invariant bleiben. Intrinsische Objekteigenschaften müssen aus der Gesamtinformation seiner zweidimensionalen Abbildung extrahiert werden. Im Falle dieser Arbeit ist das nur die Objektkontur.

Die Theorie der projektiven Invarianz existiert seit Ende des 19. Jahrhunderts und besteht aus zwei Hauptrichtungen: algebraische und differenziale Invarianz. Die Theorie der differenzialen Invarianz verwendet Ableitungen der Objektform. Der Vorteil dieses Ansatzes liegt darin, dass die Ableitungen lokal berechnet werden können und somit für Aufgaben mit teilweiser Verdeckung geeignet sind. In [35], [36] und [31] beschäftigen sich die Autoren mit der Erweiterung und Anpassung der Theorie auf ihre Umsetzung in Computer-Vision [35]. Die Theorie der algebraischen Invarianzen ist im Gegensatz dazu nur global anwendbar. Ein Überblick über die theoretischen Grundlagen und ihre Anwendung findet man in [7]. Die Arbeit bietet eine Übersicht über die Verwendung diverser sog. cross-ratios als invariante Größen bei Stereo- und Monosicht. Diese basieren auf einer Menge von Punkt-Korrespondenzen, mit deren Hilfe sich blickwinkel-invariante Konstanten ermitteln lassen. In [7] wird das Theorem über das cross-ratio vierer auf einer Geraden liegender Punkte für nicht-koplanare Punkte verallgemeinert.

Das bekannte Problem der differentialen Theorie besteht darin, dass sie höhere Ableitungen verwendet um invariante Charakteristika zu gewinnen. Rauschen der Bilddaten und die diskrete Darstellung macht eine präzise Berechnung von Ableitungen höher zweiten Grades sehr schwer. Deshalb werden häufig semi-differentiale Invarianten (z.B. [18], [8]), statt differentialer benutzt. Diese vereinigen algebraische und differentiale Theorie und erlauben es, Ableitungen höherer Ordnung zu vermeiden.

Innerhalb dieser Arbeit beschränkt man sich auf die Berücksichtigung einer Untermen-

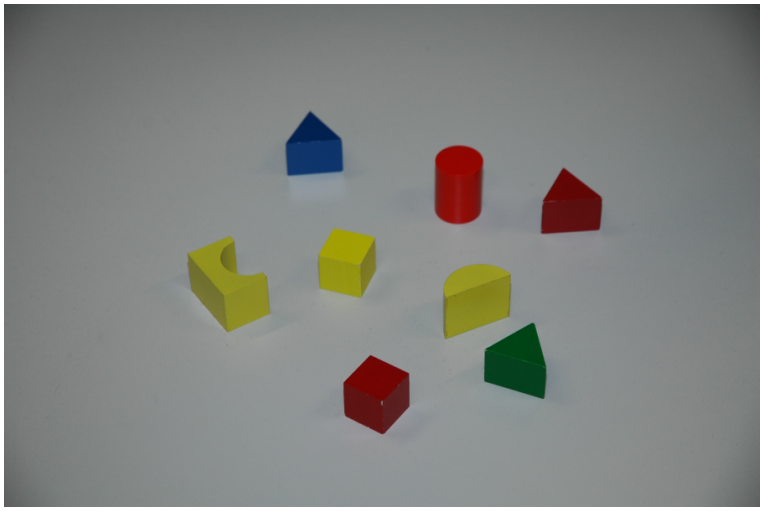


Abbildung 3: Einige Testobjekte in der korrekten Positionierung.

ge der projektiven Transformationen, den affinen Transformationen Rotation, Translation und Skalierung. Das folgt aus einer Annahme, die im COSPAL-Projekt zugrundegelegt wird. Diese besagt, dass innerhalb der experimentellen Rahmenbedingungen die Konturen der Testobjekte durch die Blickwinkelveränderungen nur gering verzerrt werden. Hierfür wird bei der Figurpositionierung die Einschränkung gemacht, dass die Objekte mit ihren klassen-charakteristischen Oberflächen nach oben zeigen müssen. Abbildung 3 illustriert innerhalb des Projektes zugelassenen Objektplazierungen. Die andere Restriktion bezieht sich auf die Objektaufnahme. Innerhalb der Experimente bewegt sich die Kamera in einem eingeschränkten Bereich ausschließlich parallel zur Arbeitsfläche. Diese Einschränkungen ermöglichen außerdem eine eindeutige Zuordnung der Figuren auf der zweidimensionalen Projektion. Konturen auf der Seite liegender oder von der Seite aufgenommener Zylinder, Brücken oder Quader können häufig auch vom menschlichen Auge nicht mehr mit Sicherheit voneinander getrennt werden.

Normierte Zernike-, pseudo Zernike-Momente und Fourier-Deskriptoren besitzen die gefragten affinen Invarianz-Eigenschaften und können erfolgreich zur Datenreduktion und in Klassifikation verwendet werden. In den nächsten Abschnitten wird sowohl die Theorie als auch die Anwendung der Fourier-Deskriptoren und Momente in der Merkmalsextraktion ausführlich beschrieben.

2.1 Fourier-Reihen und Fourier-Transformation

Die diskrete Fourier-Transformation ist eine der am häufigsten verwendeten Methoden für die Merkmalsextraktion aus Konturdaten. Zusammengefasst ergibt eine Dekomposition der diskreten Konturfunktion in eine gewichtete Summe sinusförmiger Komponenten einen komplexen Vektor A von Fourier-Deskriptoren. Der normalisierte Vektor A (Abschnitt 2.3) ist invariant gegen Rotation, Translation und Skalierung, wobei abhängig von der Anwendung häufig schon wenige seiner Elemente reichen, um eine Kontur im Klassifikationsvorgang erfolgreich zu repräsentieren. Fourier-Reihen und Fourierreihenentwicklung für die kontinuierlichen Funktionen bilden eine Grundlage für die diskreten Transformationsverfahren.

Fourier-Reihen sind nach dem französischen Wissenschaftler Joseph Fourier benannt, der sie in seiner im Jahr 1822 veröffentlichen Arbeit "Théorie Analytique de la Chaleur" benutzt hat. Sie werden heutzutage in vielen Bereichen zur Analyse periodischer Funktionen verwendet.

Sei $f : \mathbb{R} \to \mathbb{C}$ eine periodische, stückweise stetige und differenzierbare Funktion mit der Periode L. Außerdem gelte für $t_1, t_2 \in \mathbb{R}$:

$$\int_{t_1}^{t_2} |f(t)|^2 dt < \infty,$$

wobei $t_2 - t_1 = L$. Dann heisst die folgende Darstellung der Funktion f **Fourierreihenentwicklung:**

$$f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}[a_n \cos(\omega_n t) + b_n \sin(\omega_n t)], \tag{1}$$

wobei

$$\begin{aligned} \omega_n &= n\frac{2\pi}{L}, \\ a_n &= \frac{2}{L}\int_{t_1}^{t_2} f(t)\cos(\omega_n t)dt, \\ b_n &= \frac{2}{L}\int_{t_1}^{t_2} f(t)\sin(\omega_n t)dt. \end{aligned}$$

In Gleichung 1 wird die Funktion f als eine Summe reeller Kosinus- und Sinus-Terme dargestellt. Die analoge komplexe exponentiale Form lautet:

$$f(t) = \sum_{n=-\infty}^{+\infty} c_n e^{i\omega_n t},$$

wobei $i^2 = -1$ und

$$c_n = \frac{1}{L} \int_{t_1}^{t_2} f(t) e^{-i\omega_n t} dt$$

ist. Bekanntermaßen gilt:

$$e^{i\omega_n t} = \cos(\omega_n t) + i \sin(\omega_n t).$$

Die **Fourier-Transformation** der Funktion f wird folgendermaßen formuliert:

$$\Phi(p) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} f(x) e^{ipx} dx.$$

Diese Abbildung wird meistens als die Transformation der Amplitudenfunktion in den komplexen Frequenzraum veranschaulicht. Es gilt auch die Umkehrung:

$$f(x) = \int_{-\infty}^{+\infty} \Phi(p) e^{-ipx} dp.$$

Für diskrete Daten benutzt man die **finite Fourier-Transformation**. Ein Vektor komplexer Zahlen $(a_0, \ldots, a_{L-1}) \in \mathbb{C}^L$ wird in einen Vektor von **Fourier-Koeffizienten** $(A_0, \ldots, A_{L-1}) \in \mathbb{C}^L$ nach der folgenden Vorschrift transformiert:

$$A_m = \frac{1}{L} \sum_{n=0}^{L-1} a_n e^{-2\pi inm/L}. \tag{2}$$

Diese Transformation wird diskrete Fourier-Transformation(DFT) genannt und die komplexen Koeffizienten A_m heißen **Fourier-Deskriptoren (FD)**. Für die Auswertung des Summen-Terms in Gleichung 2 benötigt man $O(\mathrm{L}^2)$ arithmetische Operationen. *Fast Fourier Transform (FFT)* ist ein Algorithmus dessen Komplexität nur $O(\mathrm{L} \log \mathrm{L})$ beträgt. Innerhalb dieser Arbeit eingesetzte MATLAB-Funktion *fft* basiert auf der Software-Bibliothek FFTW [1, 16]. FFTW verwendet den Algorithmus von Cooley-Turkey [11], dessen Rechenzeit am geringsten ist, wenn L eine Zweierpotenz darstellt. Die inverse diskrete Fourier-Transformation (IDFT) wird definiert durch:

$$a_n = \sum_{m=0}^{L-1} A_m e^{2\pi inm/L}.$$

2.2 Fourier-Deskriptoren geschlossener Konturen

Sei eine Kontur C in der komplexen Darstellung mit $C : \{0, \ldots, L-1\} \to \mathbb{C}$ durch die Funktionen x und y vorgegeben (siehe Abschnitt 3.1 Gleichung 15). Für die Berechnung

der Fourier-Deskriptoren von C betrachtet man den komplexen Vektor

$$a := (a_0, \ldots, a_{L-1}) \in \mathbb{C}^L,$$

wobei für ein $k \in \{0, \ldots, L-1\}$ gilt:

$$a_k = x(k) + iy(k).$$

Hierbei entsprechen die a_ks den nacheinanderfolgenden Pixeln bzw. ihren (x, y)-Koordinaten. Durch eine Erweiterung des Definitionsbereiches (siehe Abschnitt 3.1 Gleichung 16) wird die Kontur zu einer L-periodischen Funktion auf $\mathbb{Z}$. Der Vektor a stellt dementsprechend die Funktionswerte von C im Intervall von einer Periode dar. Den komplexen Vektor von Fourier-Deskriptoren erhält man nach diskreter Fourier-Transformation von Vektor a. Anschaulich beschrieben, konvertiert die DFT bzw. FFT eine diskrete Kontur in ihr globales Formspektrum, dessen Definitionsbereich sich von rund bis eckig erstreckt. Je eckiger die Kontur ist, desto höher muss der Parameter n der Kosinus- bzw. Sinus-Terme sein, um sie annähern zu können.

In dieser Arbeit werden FD für die Merkmalsextraktion von Konturen aufgrund ihrer Invarianzeigenschaften, Berechnungsgeschwindigkeit, Kompaktheit der Datendarstellung und der guten Testergebnisse verwendet. Im nächsten Abschnitt werden die einzelnen Normalisierungsschritte der erhaltenen Fourier-Deskriptoren beschrieben.

2.3 Normalisierung der Fourier-Deskriptoren

Sei $a := (a_0, \ldots, a_{L-1}) \in \mathbb{C}^L$ der Vektor von Konturdaten und $A := (A_0, \ldots, A_{L-1}) \in \mathbb{C}^L$ der Vektor daraus berechneter Fourier-Deskriptoren. Um mit A die erwünschten invarianten Größen zu erhalten, müssen die FD einem Normalisierungsvorgang unterzogen werden. Der folgende Vektor besteht aus Beträgen aller Fourier-Deskriptoren ohne den nullten Deskriptor A_0:

$$I := (|A_1|, \ldots, |A_{L-1}|). \tag{3}$$

Bekannterweise ist I translations- und rotationsinvariant [32]. Betrachte zwei Konturen a_1 und a_2, wobei die zweite durch eine beliebige Rotation und Translation der ersten erhalten worden ist. Seien weiter die zu a_1 und a_2 berechneten Deskriptoren gegeben durch $(A_0, \ldots, A_{L-1})$ bzw. $(B_0, \ldots, B_{L-1})$. Dann gilt wegen Translations- und Rotationsinvarianz:

$$I_1 := (|A_1|, \ldots, |A_{L-1}|) = (|B_1|, \ldots, |B_{L-1}|) =: I_2. \tag{4}$$

Skaliert man die Konturen a_1 und a_2 mit beliebigen Faktoren $c_1, c_2 \in \mathbb{R}_+$, so werden die neuen Kontur-Vektoren mit a_{c1} und a_{c2} bezeichnet und die entsprechenden FD sind dann gegeben durch:

$$I_{c_1} := (|c_1 A_1|, \ldots, |c_1 A_{L-1}|) \text{ und } I_{c_2} := (|c_2 B_1|, \ldots, |c_2 B_{L-1}|).$$

Normiert man die Vektoren I_{c_1} und I_{c_2} auf eine feste Größe, z.B. auf eins mittels der Division durch die euklidische Länge:

$$I'_{c_1} = \frac{I_{c_1}}{\|I_{c_1}\|_2} \tag{5}$$

und I_{c_2} - analog, so gilt für die resultierenden Vektoren I'_{c_1} und I'_{c_2}:

$$\begin{aligned} I'_{c_1} = \frac{(|c_1 A_1|, \ldots, |c_1 A_{L-1}|)}{\sqrt{\sum_{i=1}^{L-1} |c_1 A_i|^2}} &= \frac{(|A_1|, \ldots, |A_{L-1}|)}{\sqrt{\sum_{i=1}^{L-1} |A_i|^2}} \\ &\overset{(4)}{=} \frac{(|B_1|, \ldots, |B_{L-1}|)}{\sqrt{\sum_{i=1}^{L-1} |B_i|^2}} = \frac{(|c_2 B_1|, \ldots, |c_2 B_{L-1}|)}{\sqrt{\sum_{i=1}^{L-1} |c_2 B_i|^2}} = I'_{c_2}, \end{aligned} \tag{6}$$

Mit der Normierung aus Gleichung 5 vermeidet man den Fehler, der dann entstehen kann, wenn die Größe im Nenner, normalerweise der Betrag einer von FD, nahe Null ist. Mit Gleichung 6 wurde gezeigt, dass man mit dem obigen Normalisierungsverfahren Translations-, Rotations-, und Skalierungsinvarianz erhalten kann.

2.4 Momente

Das Konzept der Momente stammt aus der Physik und wird in der Mathematik und in der Computer-Vision verwendet. Innerhalb dieser Arbeit dienen Momente wie die Fourier-Deskriptoren der Merkmalsextraktion. Ein Moment der Ordnung $(p + q)$ nehme für eine stetige Funktion $f : \mathbb{R}^2 \to \mathbb{R}$ die folgende Form an:

$$M_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \psi_{pq}(x, y) f(x, y) dx dy \qquad p, q = 0, 1, 2, \ldots \tag{7}$$

$\psi_{pq}(x, y)$ wird *weighting kernel* oder Basisfunktion genannt. In der Bildverarbeitung kann die Funktion f ein Grauwert- oder Binärbild darstellen. Dafür sei der Intensitätswert des Pixels an der Stelle (x, y) mit P_{xy} bezeichnet. Analog zur Gleichung 7 werden die Momente für eine diskrete Bildfläche definiert:

$$M_{pq} = \sum_x \sum_y \psi_{pq}(x, y) P_{xy} \qquad p, q = 0, 1, 2, \ldots$$

Abhängig von der Basisfunktion wird zwischen zwei Gruppen der Momente unterschieden, den orthogonalen und nicht-orthogonalen. Orthogonalität für zwei Funktionen $y_n : A \to \mathbb{R}$ und $y_m : A \to \mathbb{R}$, $n, m \in \mathbb{N}$, $A \subseteq \mathbb{R}$ ist definiert durch:

$$y_n \perp y_m : \quad \Leftrightarrow \quad \int_{x \in A} y_n(x) y_m(x) dx = 0.$$

Im diskreten Fall kann dieses Integral durch eine entsprechende Summe ersetzt werden.

In [34] werden sechs Arten der Momente untersucht: geometrische, Legendre-, Zernike-, pseudo Zernike-, komplexe und Rotationsmomente. Die Autoren führen Tests in drei Bereichen durch: Empfindlichkeit für Rauschen, Informationsredundanz, und Bild-Repräsentationskapazitäten. Die Tests wiesen nach, dass die orthogonale Gruppe (Zernike, Legendre und pseudo-Zernike) bessere Ergebnisse im Vergleich zu der nicht-orthogonalen Gruppe liefert. Es wird in [34] nach den durchgeführten Untersuchungen in drei Bereichen zusammengefasst, dass Zernike- und pseudo Zernike-Momente andere Momente in ihrer Gesamtleistung übertreffen.

2.4.1 Zernike- und pseudo Zernike-Momente

Zernike-Polynome wurden im Jahr 1934 von dem holländischen Physiker Frederik Zernike in [39] vorgestellt. Sie spielen eine wichtige Rolle im Feld der geometrischen Optik. Die Menge der komplexen Zernike-Polynome ist orthogonal innerhalb des Einheitskreises und wird folgendermaßen definiert:

$$V_{nm}(x, y) = V_{nm}(\rho, \theta) = R_{nm}(\rho) \exp(im\theta), \tag{8}$$

wobei wieder $i^2 = -1$, $n \in \mathbb{N}_0$, $m \in \mathbb{Z}$ mit der Einschränkung: $n - |m|$ gerade ist und $|m| \leq n$ mit $\rho = \sqrt{x^2 + y^2}$. θ ist der Winkel zwischen dem Vektor ρ und der x-Achse in der Richtung gegen den Uhrzeigersinn. Sog. radiale Polynome $R_{nm}(\rho)$ werden definiert durch:

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-s)!}{s! \left(\frac{n+|m|}{2} - s\right)! \left(\frac{n-|m|}{2} - s\right)!} \rho^{n-2s}. \tag{9}$$

Die Menge der Zernike-Polynome wird als Basisfunktion bei der Erzeugung der Zernike-Momente (ZM) verwendet. Sei $f : \mathbb{R}^2 \to A$ mit $A = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 \leq 1\}$, dann kann die Basistransformation nach der folgenden Vorschrift durchgeführt werden:

$$Z_{nm} = \frac{n+1}{\pi} \int_{(x,y) \in A} f(x, y) [V_{nm}(\rho, \theta)]^* dx dy, \tag{10}$$

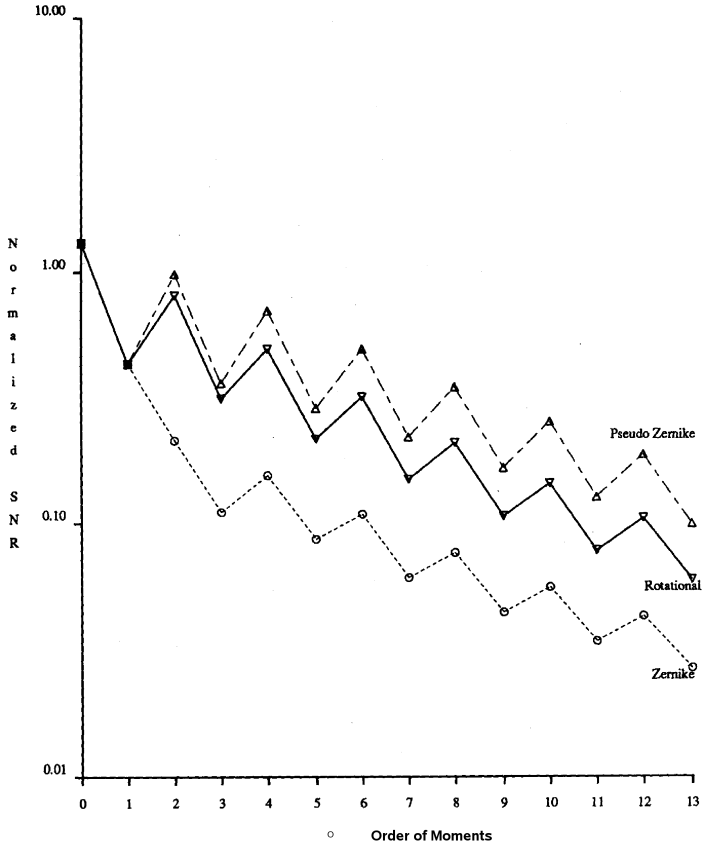


Abbildung 4: Ein Vergleich normalisierter Signal-Rausch-Verhältnisse für pseudo Zernike-, Rotations- und Zernike-Momente. Die Y-Achse ist logarithmisch skaliert. Grafik ist entnommen aus [34].

wobei * für die komplexe Konjugation steht. Im Falle einer diskreten Bildfunktion wird in Gleichung 10 das Intergral durch eine Doppelsumme ersetzt:

$$Z_{nm} = \frac{n+1}{\pi} \sum_x \sum_y P_{xy}[V_{nm}(\rho, \theta)]^*, \tag{11}$$

wobei weiterhin gilt: $x^2 + y^2 \leq 1$. Zur Umkehrtransformation siehe [21].

Im Jahr 1954 wurde von Bhatia und Wolf [38] die folgende Menge von Polynomen vorgestellt:

$$R'_{nm}(\rho) = \sum_{s=0}^{n-|m|} (-1)^s \frac{(2n+1-s)!}{s!(n-|m|-s)!(n+|m|+1-s)!} \rho^{n-s},$$

mit $n \in \mathbb{N}_0$, $m \in \mathbb{Z}$ mit $|m| \leq n$. Verwendet man in Gleichung 8 diese Vorschrift, so erhält man die Menge der Polynome, die pseudo Zernike-Polynome genannt werden. Die pseudo Zernike-Momente werden mit Hilfe der pseudo Zernike-Polynomen analog zur Gleichung 10 definiert. Sie besitzen die gleichen Invarianz-Eigenschaften wie die Zernike-Momente. In Abbildung 4 entnommen aus [34] kann man leicht erkennen, dass pseudo Zernike-Momente (PZM) sich deutlich besser als Zernike- und Rotationsmomente unter Rauscheinfluss verhalten.

2.4.2 Effiziente Berechnungsmethode

Die Berechnung von Momenten ist sehr rechenintensiv. In [4] wird eine Methode vorgestellt, die es erlaubt, durch die Identifikation von wiederholt verwedeten Termen Momente höherer Ordnung ohne jeglichen Verlust an Genauigkeit effizienter zu berechnen.

Dieser Algorithmus kann sowohl für Zernike- als auch für pseudo Zernike-Momente verwendet werden. In [4] wird er für Zernike-Momente beschrieben. Die Gleichung 9 wird in [4] folgendermaßen umgeformt:

$$R_{nm} = \sum_{k=|m|,n-k=\text{even}}^{n} = \frac{(-1)^{\frac{n-k}{2}} \frac{n+k}{2}!}{\frac{n-k}{2}! \frac{k+m}{2}! \frac{k-m}{2}!} \rho^k = \sum_{k=|m|,n-k=even} \beta_{nmk} \rho^k.$$

Weiter folgt nach [4]:

$$\begin{aligned} Z_{nm} &= \frac{n+1}{\pi} \sum \sum_{x^2+y^2 \leq 1} \left(\sum_{k=|m|}^{n} \beta_{nmk} \rho^k \right) \exp(-jm\theta) f(x,y) \\ &= \frac{n+1}{\pi} \sum_{k=|m|}^{n} \beta_{nmk} \left(\sum \sum_{x^2+y^2 \leq 1} \exp(-jm\theta) \rho^k f(x,y) \right) \\ &= \frac{n+1}{\pi} \sum_{k=|m|}^{n} \beta_{nmk} \chi_{mk}. \end{aligned}$$

Der Vorteil dieser Darstellung wird aus der Abbildung 5 sofort ersichtlich. Bei der Implementierung können die wiederholt vorkommenden Terme einmal berechnet werden und für die spätere Verwendung abgespeichert werden. In [4] findet man einen Vergleich vom Rechenaufwand dieser Methode mit den anderen effizienten Methoden. Die von den Autoren in [4] vorgeschlagene Methode wird hier vorgezogen, da sie keine Genauigkeitsverluste mit sich zieht.

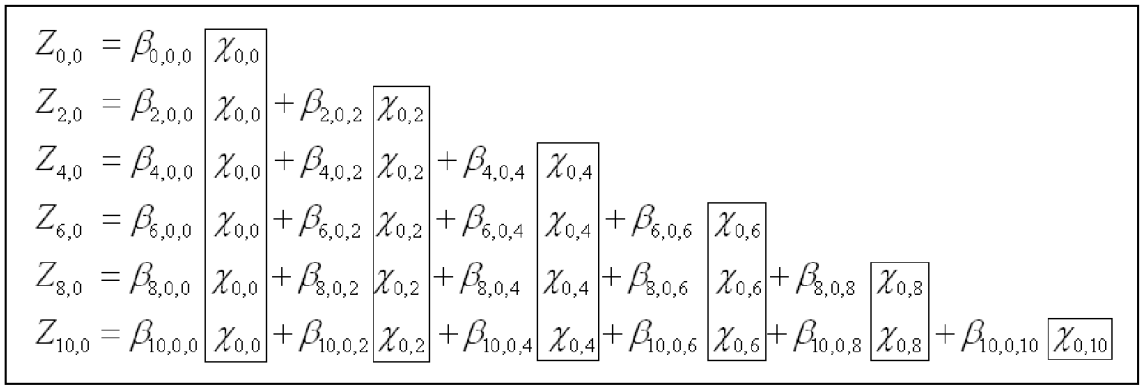


Abbildung 5: Gemeinsame Terme bei der Berechnung von Zernike-Momenten; Tabelle ist aus [4] entnommen.

2.5 Normalisierung der Zernike- und pseudo Zernike-Momente

Genau wie die FD müssen ZM und PZM einem Normalisierungsvorgang unterzogen werden, um invariante Eigenschaften zu erzielen. Zernike- und pseudo Zernike-Momente sind komplexe Zahlen, deren Längen bzw. Beträge rotationsinvariante Größen darstellen [21]. Allerdings sind sie weder translations- noch skalierungsinvariant.

Zentrierte Momente sind nach [21] translationsinvariant. Um diese zu erhalten, verschiebt man den Schwerpunkt der ursprünglichen Funktion auf Null:

$$\hat{f}(x, y) = f(x + \bar{x}, y + \bar{y}), \tag{12}$$

wobei $\bar{x}$ und $\bar{y}$ mit Hilfe der Momente m_{pq} der nullten und ersten Ordnung berechnet werden können:

$$\bar{x} = \frac{m_{10}}{m_{00}} \text{ und } \bar{y} = \frac{m_{01}}{m_{00}}.$$

m_{pq} wird nach der folgenden Gleichung ermittelt:

$$m_{pq} = \sum_x \sum_y x^p y^p f(x, y).$$

Die Skalierungsinvarianz für Konturdaten folgt, indem man die Anzahl der Pixel auf eine vorgegebene konstante Größe β normiert [21].

Zusammengefasst werden für die Berechnung der Momente und ihre Normalisierung die folgenden Schritte durchgeführt: die Konturfunktion wird als Erstes auf den Einheitskreis abgebildet. Dafür wird ein passender Skalierungsfaktor ermittelt. Innerhalb des

Einheitskreises sind die Zernike- bzw. pseudo Zernike-Polynome orthogonal. Der Kontur-Schwerpunkt muss in den Koordinatenursprung verschoben werden, um die Translationsinvarianz zu garantieren. Letztendlich muss die Anzahl der Pixel auf einen festen Wert gesetzt werden (z.B. $\beta = 64$). Dieser Schritt wird mit Hilfe von Subsampling oder einer Erhöhung der Sampling-Rate durchgeführt. Bildet man die Summe in der Gleichung 11 über eine unterschiedliche Anzahl der diskreten Punkte, abhängig von der Sampling-Rate, so sind die erhaltenen Werte nicht vergleichbar. In [21] wird gezeigt, dass die Beträge der so berechneten Momente die gewünschten Eigenschaften der Rotations-, Translations-, und Skalierungsinvarianz besitzen.

Durch den Normierungsvorgang kommt es dazu, dass $|Z_{00}|$ und $|Z_{11}|$ für alle Konturen feste Größen aufweisen und deshalb nicht mehr in der Klassifikation verwendet werden können. $|Z_{11}|$ ist immer Null und für $|Z_{00}|$ folgt direkt aus Gleichung 10:

$$|Z_{00}| = \frac{\beta}{\pi}.$$

Im experimentellen Teil (Abschnitt 6.2.3) werden Momente der Ordnung nicht größer als 9 verwendet.

3 Konturdarstellung und -vorverarbeitung

3.1 Darstellung

Sei $[a, b] \subset \mathbb{R}$, dann ist eine Kontur nach [20] eine auf dem Intervall $[a, b]$ stückweise stetig differenzierbare, einfache, geschlossene Kurve. Weiterhin wird eine Kurve in einer Parameterdarstellung folgendermaßen definiert:

$$\Gamma = \{(x(t), y(t)) | t \in [a, b]\}, \tag{13}$$

wobei x und y reellwertige, stetige Funktionen mit Definitionsbereich $[a, b]$ sind.

In der Computer-Vision existieren viele Möglichkeiten eine diskrete Kontur darzustellen. Einige davon sind: cross-section Funktion, Radius-Vektor Funktion, Breitenfunktion, parametrisierte und komplexe Funktion, Tangente-Winkel Funktion und Krümmungsfunktion [22]. Für diese Arbeit erwiesen sich die komplexe und die parametrisierte Konturdarstellung als am geeignetsten.

Die parametrisierte Darstellung einer Kontur mit L Punkten sieht folgendermaßen aus:

$$c : \{0, \ldots, L-1\} \to \mathbb{R} \times \mathbb{R}, \quad l \mapsto (x(l), y(l)). \tag{14}$$

Die obige Darstellung wird interpretiert als Bewegung auf der Kontur ab dem Ausgangspunkt $(x(0), y(0))$ von einem Punkt zum nächsten.

Analog folgt die parametrisierte Konturdarstellung in Polar-Koordinaten. Sei dafür $I := [0, 2\pi]$ und:

$$p : \{0, \ldots, L-1\} \to \mathbb{R} \times I, \quad l \mapsto (\rho(l), \theta(l)),$$

wobei für den gegebenen Koordinatenursprung (x_0, y_0) und die Euklidische Metrik d gilt:

$$\rho : \{0, \ldots, L-1\} \to \mathbb{R}, \quad l \mapsto d((x_0, y_0), (x(l), y(l)))$$

und

$$\theta : \{0, \ldots, L-1\} \to I, \quad l \mapsto \begin{cases} \pi + \arctan(y(l)/(x(l)), & \text{falls } x(l) < 0 \\ \arctan(y(l)/x(l)), & \text{falls } x(l) > 0,\ y(l) \geq 0 \\ \arctan(y(l)/x(l)) + 2\pi, & \text{falls } x(l) > 0,\ y(l) < 0 \\ \frac{1}{2}\pi, & \text{falls } x(l) = 0,\ y(l) > 0 \\ \frac{3}{2}\pi, & \text{falls } x(l) = 0,\ y(l) < 0, \end{cases}$$

die erweiterte Version für die resultierenden Winkel aus dem Intervall $[0, 2\pi]$ darstellt.

Die dritte verwendete Konturdarstellung in dieser Arbeit ist die komplexe:

$$C : \{0, \ldots, L-1\} \to \mathbb{C}, \quad l \mapsto x(l) + iy(l) \tag{15}$$

Die obigen Funktionen c, p und C haben die gleiche Bedeutung und unterscheiden sich nur durch die Verschiedenheit der Koordinatensysteme.

Alle drei oben beschriebenen Funktionen können als periodische Funktion mit der Periodenlänge gleich L aufgefasst werden. Dies geschieht durch eine Erweiterung des Definitionsbereichs auf $\mathbb{Z}$:

$$c(n + pL) = c(n) \tag{16}$$

für $p \in \mathbb{Z}$ und $n \in \{0, \ldots, L-1\}$.

3.2 Krümmungsfunktion und Eckpunkte

Sei eine kontinuierliche Kontur in der parametrischen Darstellung durch die Funktionen x und y (siehe Gleichung 13) vorgegeben. Dann ist die Krümmung im Punkt $p := (x(t), y(t))$, $t \in [a, b]$ definiert durch:

$$k(p) = \frac{x'(t)y''(t) - x''(t)y'(t)}{\left(x'^2(t) + y'^2(t)\right)^{3/2}}, \tag{17}$$

falls in p die ersten und die zweiten Ableitungen von x und y existieren. Sonst ist die Krümmung undefiniert. Der Betrag von $k(p)$ gibt die Geschwindigkeit der Richtungsänderung der Tangente im Punkt p wieder.

Mit *points of interest (POI)* bezeichnet man die Bildpunkte, die eine kennzeichnende lokale Eigenschaft besitzen [6] und die trotz perspektivischer Transformationen oder der Veränderung der Lichtverhältnisse stabil ermittelt werden können. Für die Arbeit mit Konturen verwendet man häufig den Begriff *Eckpunkt* im Sinne von points of interest. Eine der Möglichkeiten, Eckpunkte einer Kontur zu definieren, ist über die Krümmungswerte, wie beispielsweise in [22]. Ein Eckpunkt ist dann ein Punkt, in dem die Krümmung undefiniert ist. Diese Definition ermöglichen es, die gleiche Menge der Eckpunkte unabhängig von der Rotation, Skalierung oder Translation einer Kontur zu extrahieren.

Im diskreten Fall müssen einige Anpassungen bei der Ermittlung der Eckpunkte gemacht werden. Sei eine Kontur c in der parametrischen Darstellung (siehe Gleichung 14) durch die Funktionen x und y vorgegeben. Für die Berechnung der ersten und der zweiten

Ableitung werden wie in [24] die folgenden Vorschriften verwendet:

$$\begin{aligned} x'(l) &= x(l+1) - x(l-1), \\ y'(l) &= y(l+1) - y(l-1), \\ x''(l) &= x(l+1) + x(l-1) - 2x(l), \\ y''(l) &= y(l+1) + y(l-1) - 2y(l). \end{aligned} \tag{18}$$

Hierbei müssen die entstehenden Probleme, die mit Rauschen verbunden sind, bekämpft werden. Mit der Gleichung 18 lässt sich die Krümmung in jedem Punkt berechnen. Laut [22] besitzt k Translations- und Rotationsinvarianz. Skalierungsinvarianz kann mit Hilfe einer Normalisierung des Definitionsbereichs erzielt werden.

Für die Ermittlung der Konturеckpunkte im diskreten Fall kann die Definition der Eckpunkte einer kontinuierlichen Kontur nicht mehr eingesetzt werden. Hier nimmt man an, dass ein Eckpunkt in einer lokalen Umgebung den maximalen Wert vom Betrag der Krümmungsfunktion k aufweist. Für ein $\epsilon \in \mathbb{N}$, $l \in \{0, \ldots, L-1\}$ und eine Umgebung $U_0 = U(l, \epsilon)$ ist $p_0 = (x(l_0), y(l_0))$ dann ein Eckpunkt, wenn für alle Punkte $P_u = \{(x(l), y(l)) | l \in U_0\}$ gilt:

$$|k(p_0)| = \max_{p \in P_u} |k(p)|.$$

Somit hängt im diskreten Fall die Menge der ermittelten Eckpunkte von der Wahl der lokalen Umgebung ab. Deshalb kann man hier, trotz der Invarianzeigenschaften der Krümmungsfunktion, nicht mehr garantieren, dass die Menge der Eckpunkte unter affinen Transformationen invariant bleibt.

3.3 Rauschen

In dieser Arbeit basiert das Klassifikationsverfahren auf Konturdaten, die aus den RGB-Bildern entsprechender Objekte gewonnen werden. Fehler bei der Formerkennung und die Unregelmäßigkeiten der extrahierten Konturen haben mehrere mögliche Ursprünge. Sie können während der Aufnahme zustande kommen oder auch durch die Bildverarbeitung verursacht werden.

Bildrauschen entspricht den Störungen, die keinen Bezug zum eigentlichen Bildinhalt haben. *Dunkelrauschen* entsteht durch zufällig angeregte Elemente in CCD- bzw. CMOS-Sensoren infolge der Wärme. *Ausleserauschen* wird durch das Rauschen des Ausleseverstärkers verursacht. *Digitalisierungsrauschen* entsteht dann, wenn die kontinuierlichen

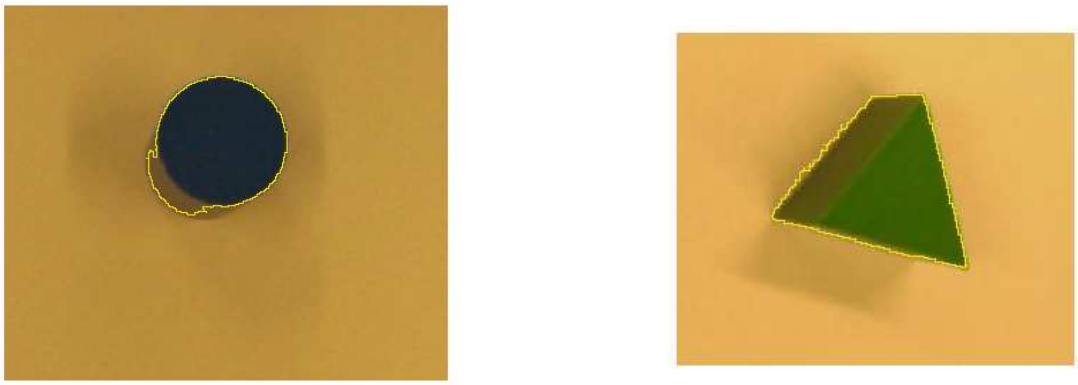


Abbildung 6: Probleme bei der Kontur-Extraktion verursacht durch Glanzlicht (links) oder Schatten (rechts).

Signale des Bildsensors in diskrete Werte umgewandelt werden.

Andere Quelle für Störungen sind die Bildinhalte, verursacht durch die ungünstigen Lichtverhältnisse bei der Bildaufnahme, Schatten oder Glanzlicht. Sie bewirken, dass man besonders im Grauwert-Bild das Objekt vom Hintergrund schlecht unterscheiden kann. Um die Kontur aus einem Bild zu extrahieren, wird hier das RGB-Bild zuerst in ein Grauwert- und anschließend in ein Schwarzweißbild umgewandelt. Infolge der oben beschriebenen Probleme kann es dazu kommen, dass bei der Grauwert-Schwarzweiss-Konvertierung manche Hintergrund-Pixel zum Objekt hinzukommen oder auch dass manche Objekt-Pixel ausgeschlossen werden. Beispiele von zwei in solchen Fällen resultierenden Konturen sieht man auf der Abbildung 6.

Um Effekte des Rauschens auf die zukünftigen Berechnungen möglichst niedrig zu halten, wird im nächsten Abschnitt ein Verfahren zur Konturvorverarbeitung vorgeschlagen, das die Konturdaten von verschiedenen Rauschfaktoren zu bereinigen sucht.

3.4 Vorverarbeitung

Zu den Hauptaufgaben der Konturvorverarbeitung in dieser Arbeit gehört Rauschenbeseitigung, Datenreduktion und Formatierung für die Merkmalsextraktion. Das zweidimensionale Bild einer Figur (wie beispielsweise links in Abbildung 7) dient als Ausgangspunkt für die Konturextraktion. Die Qualität der aus einem RGB-Bild gewonnenen rohen Kon-

turdaten ist durch mehrere physikalisch- und algorithmisch-verursachte Rauschfaktoren (Abschnitt 3.3) beeinflusst. Die erste Aufgabe der Vorverarbeitung liegt deshalb darin, die rohen Daten zu entrauschen. Dazu gehört nach Möglichkeit die Beseitigung von kleinen Konturdellen oder Hügeln, die durch Glanzlicht bzw. Schatten entstehen. Nach der Extraktion aus dem Schwarzweißbild besitzt die Kontur lokal eine zackige Struktur (siehe die Abbildung 7 rechts). Für eine angemessene Berechnung der Merkmale ist es hilfreich, diese durch die RGB-Schwarzweißkonvertierung zustande gekommene lokale Struktur dem glatten Verlauf der Originalform anzunähern. Die andere Aufgabe der Vorverarbeitung besteht darin, die Daten für den Merkmalsextraktor in den richtigen Format zu bringen. Dazu gehört z.B. subsampling auf die erforderliche Pixel-Anzahl.

Modellierung mit kubischen B-Splines wird hier erstens für die Korrektur der oben beschriebenen lokalen Strukturunregelmäßigkeiten eingesetzt. Gleichzeitig trägt diese Methode zu einer erheblichen Datenreduktion bei. Die globale Konturstruktur bleibt dabei dennoch erhalten. Zweitens wird die Menge der B-Spline Stützstellen später für die hierarchische Fragmentierung verwendet.

In den folgenden Abschnitten werden die einzelnen Schritte des Verfahrens, die Konturextraktion und das Glätten, die Segmentierung und letztendlich die Modellierung detaillierter beschrieben.

3.4.1 Konturextraktion

Die Konturdaten werden aus den RGB-Bildern der fünf Figuren gewonnen. Hierbei wird das Bild erst in ein Grauwertbild umgewandelt. Die MATLAB-Funktion *rgb2gray* benutzt dafür den NTSC-Farbraum, in dem die Luminanz ein Grauwert-Signal darstellt und die anderen Komponenten die Sättigung und den Farbton bestimmen. Die Funktion konvertiert zuerst die RGB- in NTSC-Werte, setzt die Farb- und die Sättigungskomponenten auf null und konvertiert anschließend die erhaltenen NTSC Werte zurück in den RGB-Raum. Mit der MATLAB-Funktion *im2bw* wird das Grauwertbild in ein schwarz-weißes umgewandelt. Das Verfahren verwendet einen Schwellwert, um das Grauwertbild in ein binäres zu konvertieren. Der Schwellwert für die Konvertierung wird zur Zeit experimentell ermittelt. Die Kontur wird anschließend aus dem Schwarzweißbild mit Hilfe des MATLAB-Verfahrens *bwboundaries* extrahiert, wobei für jeden Punkt eine 8-Nachbarschaft zugelassen ist. Beispiele für ein Prisma, einen Zylinder und die daraus ergebenen Konturen sind in Abbildung 7 zu sehen.

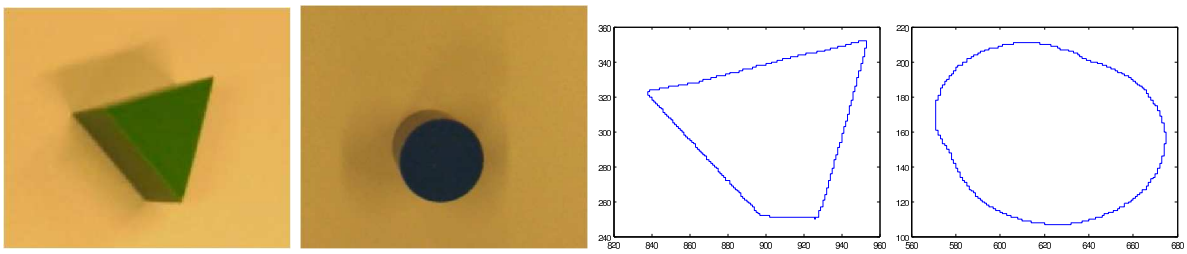


Abbildung 7: RGB-Bilder und die daraus extrahierten Konturen; Schwellwert für die Konvertierung ins Schwarzweißbild beträgt 0.48 bzw. 0.4.

3.4.2 Modellierung mit B-Splines

Im Allgemeinen besteht das Modellierungsverfahren aus zwei Schritten. Im ersten Schritt wird für die zu modellierende Funktion ein Menge Stützstellen ermittelt. Diese werden im zweiten Schritt mit Hilfe einer Menge kubischer Splines interpoliert. Bei einer geeigneten Auswahl der Stützstellen kann die Interpolation eine gute globale Näherung an die ursprüngliche Funktion liefern. Gleichzeitig werden alle lokale Strukturunregelmäßigkeiten der Originalfunktion behoben. Die resultierende Funktion ist außerdem auf dem ganzen Wertebereich zweimal stetig differenzierbar. Formal ist ein B-Spline des n-ten Grades definiert durch:

$$S : [t_0, t_m] \to \mathbb{R}^2 \quad t \mapsto \sum_{i=0}^{m-n-1} P_i b_{i,n}(t), t \in [t_0, t_m],$$

wobei

$$t_0 \leq t_1 \leq \ldots \leq t_m.$$

Mit P_i werden die Kontrollpunkte bezeichnet. Die Koeffizienten $b_{m,n}$ können mit der folgenden rekursiven Formel berechnet werden:

$$b_{j,0}(t) := \begin{cases} 1, & \text{falls} \quad t_j \leq t \leq t_{j+1} \\ 0 & \text{sonst} \end{cases}$$

$$b_{j,n}(t) := \frac{t - t_j}{t_{j+n} - t_j} b_{j,n-1}(t) + \frac{t_{j+n+1} - t}{t_{j+n+1} - t_{j+1}} b_{j+1,n-1}(t)$$

Für die Modellierung einer Kontur mit kubischen B-Splines muss eine passende Menge Eckpunkte bzw. Kontrollpunkte ermittelt werden. Der im Abschnitt 3.4.3 beschriebene Algorithmus sorgt dafür, dass die Kontrollpunkte so gewählt werden, dass die Interpolation eine gute Annäherung der globalen Formstruktur zurückgibt. Folglich enthält die Menge

der Kontrollpunkte die Konturstellen, die für die Beschreibung globaler Konturstruktur bedeutend sind. Dementsprechend können diese Punkte für die hierarchische Fragmentierung gut verwendet werden.

3.4.3 Kontursegmentierung und Interpolation

Kontursegmentierung hat innerhalb dieser Arbeit zwei Verwendungszwecke. Zum einen stellt sie eine passende Menge der Kontrollpunkte für die Interpolation mit kubischen B-Splines zur Verfügung. Zum anderen wird die gleiche Menge später in der hierarchischen Fragmentierung benutzt, um Trainingsdaten bzw. Kontursegmente für verschiedene Klassifikationsebenen zu generieren.

Das Verfahren der Kontursegmentierung besteht aus mehreren Schritten. Am Anfang wird die Kontur wie beispielsweise bei [24] mit einem Gauß-Kernel geglättet. Dies bewirkt eine Verbesserung der lokalen Struktur, die sich empirisch als hilfreich bei der Berechnung der diskreten Ableitungen erwiesen hat. Im zweiten Schritt werden die Krümmungswerte für jeden diskreten Punkt bestimmt, anhand dessen später die Kontrollpunkte bzw. Stützstellen für die Interpolation ermittelt werden.

Glätten mit Gauß-Filter

Sei c eine Kontur in der parametrischen Darstellung und L die Anzahl der diskreten Punkte, $l \in \{0, \ldots, L-1\}$. Seien weiter $k \in \{0, \ldots, K-1\}$ und $\sigma > 0$ konstant, dann ist

$$g(k) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-k^2/2\sigma}$$

der innerhalb dieser Arbeit zum Glätten der Kurve verwendeter Gauß-Kernel. Die geglättete Kontur wird in einem Punkt l definiert durch:

$$\hat{c}(l) = ((x * g)(l), (y * g)(l)), \tag{19}$$

wobei für zwei Funktionen x und g der $*$-Operator folgendermaßen definiert ist:

$$(x * g)(l) = \sum_j x(j)g(l-j).$$

In Gleichung 19 wird die Kontur mit dem Gauß-Kernel gefaltet, so dass eine Tiefpassfilterung bewirkt wird. Hierbei sollen die lokalen Artefakte, die bei der Konturextraktion zustande kommen - mindestens zum Teil - beseitigt werden. Der Wert von σ spielt eine

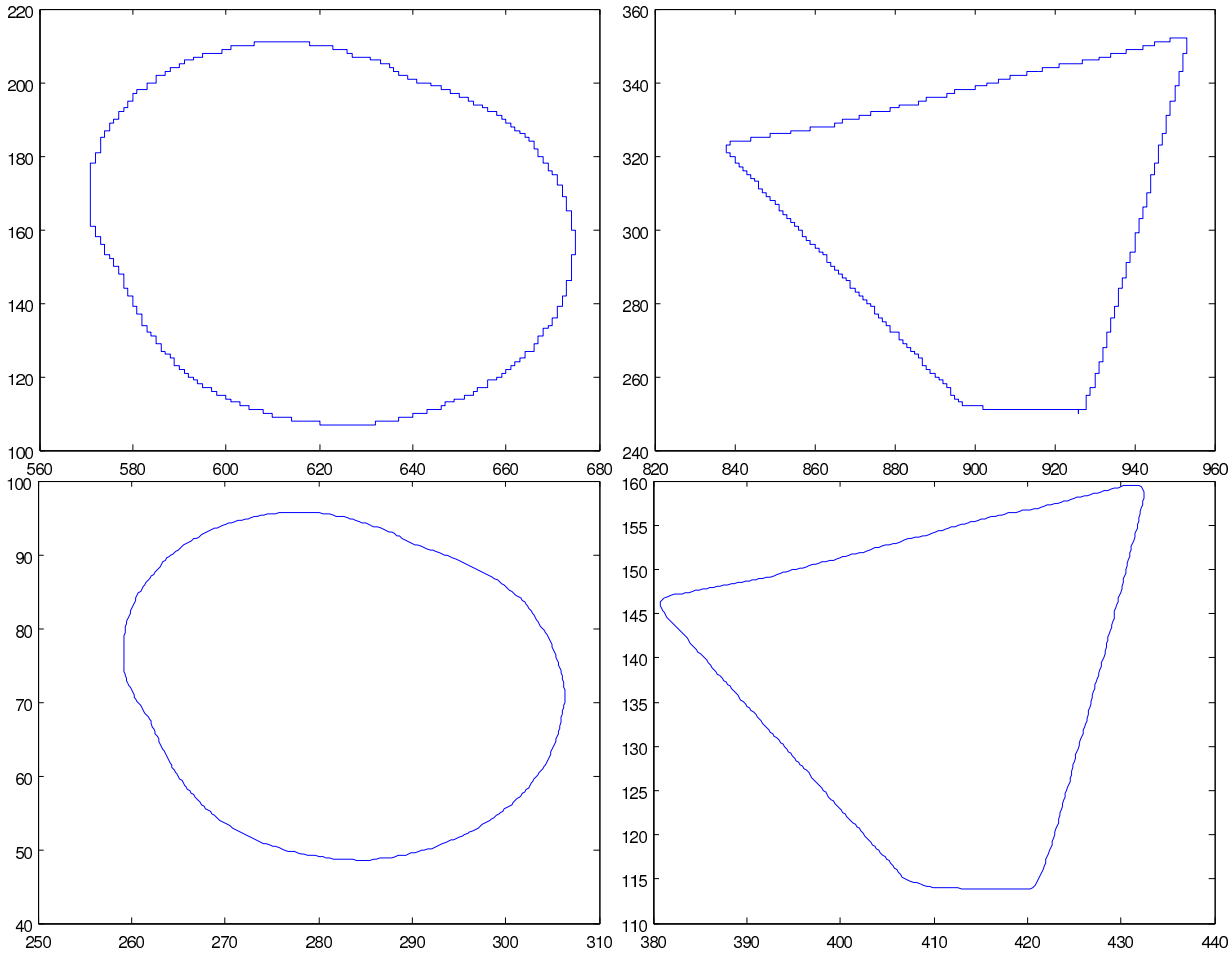


Abbildung 8: Beispiel der Glättung einer Zylinder- und einer Prisma-Kontur mit $\sigma = 5$. Originalkonturen (oben), die geglätteten Konturen (unten).

wichtige Rolle. Umso größer σ, desto glatter wird die Ergebniskontur. Zu große σ-Werte können eine eckige Form in eine nahezu ellipsenförmige verwandeln, im Gegegsatz hat bei einem kleinen σ die Glättung kaum sichtbare Auswirkungen.

Die Verwendung der Gauß-Glättung hat sich empirisch als hilfreich erwiesen (siehe Abschnitt 3.4.3). Der Wert für σ wird wiederum experimentell ermittelt. Die Abbildung 8 demonstriert ein Beispiel der Glättung mit $\sigma = 5$. In der oberen Reihe sieht man die Originalkonturen eines Prismas und eines Zylinders, in der unteren Reihe kann man die entsprechenden Konturen nach der Glättung beobachten. Während die lokale Struktur dieser Formen stark verbessert ist, unterscheidet sich bei beiden die globale Struktur kaum vom Original.

Ermittlung diskreter Ableitungen und Krümmungswerte

Empirisch wurde ermittelt, dass die Modellierung mit B-Splines eine Verbesserung der Klassifikationsgenauigkeit bewirkt. Wie oben beschrieben, erfordert dieses Verfahren eine Menge der Kontrollpunkte, die in dieser Arbeit mit Hilfe der Krümmungswerte (siehe die Gleichung 17) erworben werden.

Für die Ermittlung der Krümmung in einem diskreten Konturpunkt benötigt man die Werte der ersten und der zweiten Ableitungen. Die diskreten Ableitungen werden gewöhnlich nach der Gleichung 18 berechnet, wobei hierfür höchstens die Koordinaten von drei Nachbarpunkten in die Berechnungen mit einbezogen werden. Die rohen Konturdaten bestehen aus einer Folge ganzzahliger Pixel-Koordinaten, wobei für jeden Pixel nur acht ganzzahliger Positionstupel aus seiner direkten Umgebung als mögliche Nachbarn in Frage kommen. Betrachtet man den Wertebereich der Funktionen c' und c'' für die x- oder y-Komponente der Kontur, so sieht man sofort, dass er stets aus sechs bzw. drei gleichen Werten besteht, ganz unabhängig von der Kontur. Dies demonstriert die Abbildung 9 (oben). Die aus den rohen Daten berechneten Werte der Krümmungsfuntkion sind in Abbildung 10 abgebildet. Aus den oberen Überlegungen, unterstützt von Abbildung 10, folgt, dass die anhand der rohen Daten gewonnenen Krümmungswerte nicht der Extraktion von POI bzw. der Stützstellen dienen können.

Um die Werte der ersten und der zweiten Ableitung zu präzisieren, hat sich die Glättung empirisch als hilfreich erwiesen. In Abbildung 9 (Mitte) sind die Ableitungswerte nach der Konturglättung dargestellt. Anhand dieser Graphik kann man sofort erkennen, dass man mit der Glättung viel genauere Information über den Verlauf der Kontur gewinnt. Die damit ermittelten Krümmungswerte kann man in Abbildung 11 sehen. Trotz einer deutlichen Verbesserung im Vergleich zum ersten Ergebnis, sind die Werte an manchen Eckpunkten immer noch nicht erheblich höher als an den Kanten. Dementsprechend sind die Krümmungswerte nocht nicht für die Extraktion der POI geeignet.

Für die Verbesserung der Ableitungswerte wird hier eine Erweiterung bei der Ermittlung der ersten und der zweiten Ableitung verwendet. Diese besteht darin, die Ableitung für den jeweiligen Punkt als Mittelwert der Ableitungen für die Punkte in seiner Umgebung

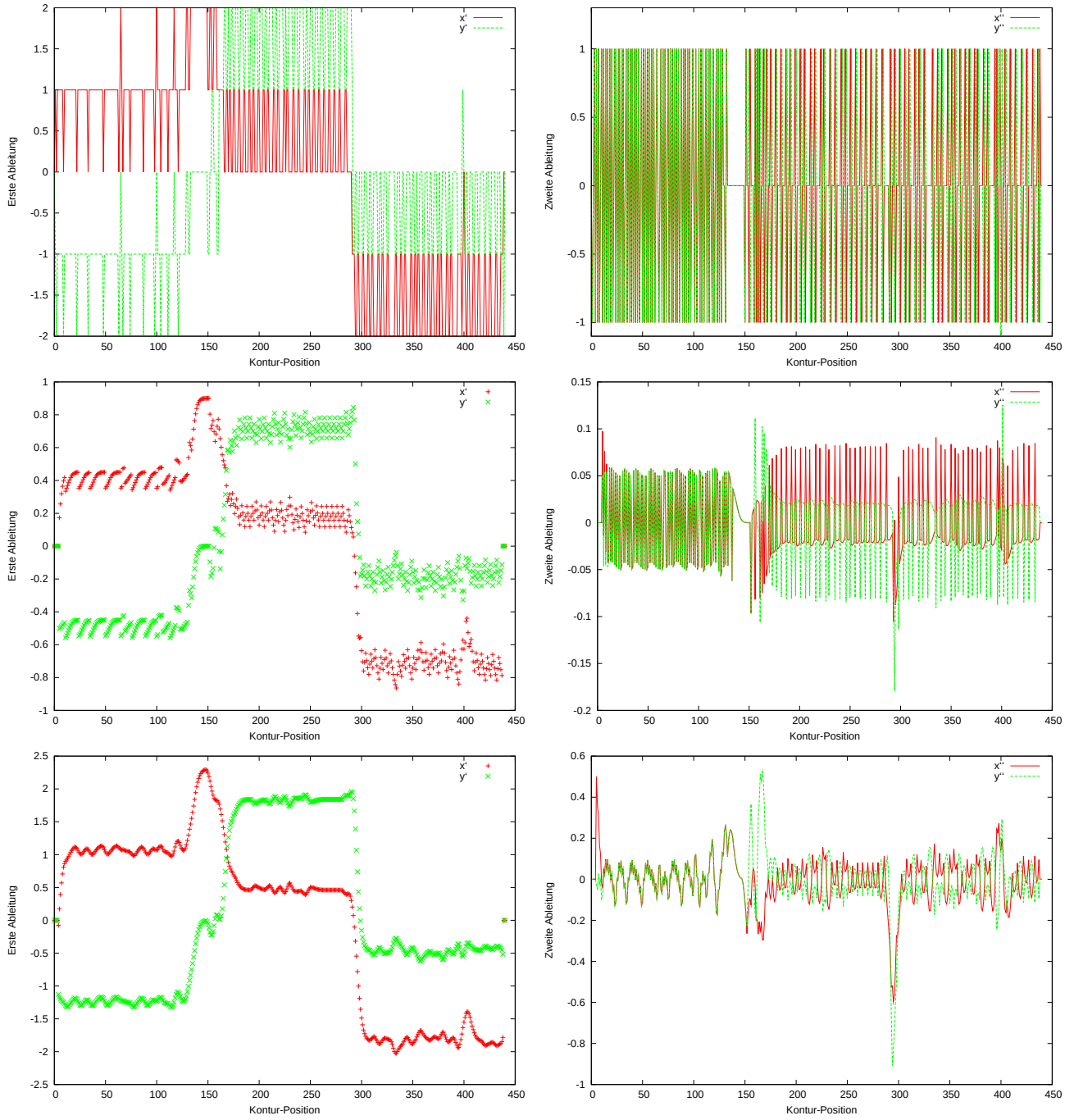


Abbildung 9: Obere Reihe: erste (links) und zweite (rechts) Ableitungen der x- und y-Konturkomponenten. Mittlere Reihe: erste (links) und zweite (rechts) Ableitungen der x- und y-Konturkomponenten, berechnet nach der Glättung der Kontur mit Gauß-Filter, $\sigma = 5$. Untere Reihe: erste (links) und zweite (rechts) über eine Umgebung mit $\epsilon = 5$ gemittelte Ableitungen der x- und y-Konturkomponenten, berechnet nach dem Glätten mit Gauß-Filter, $\sigma = 5$.

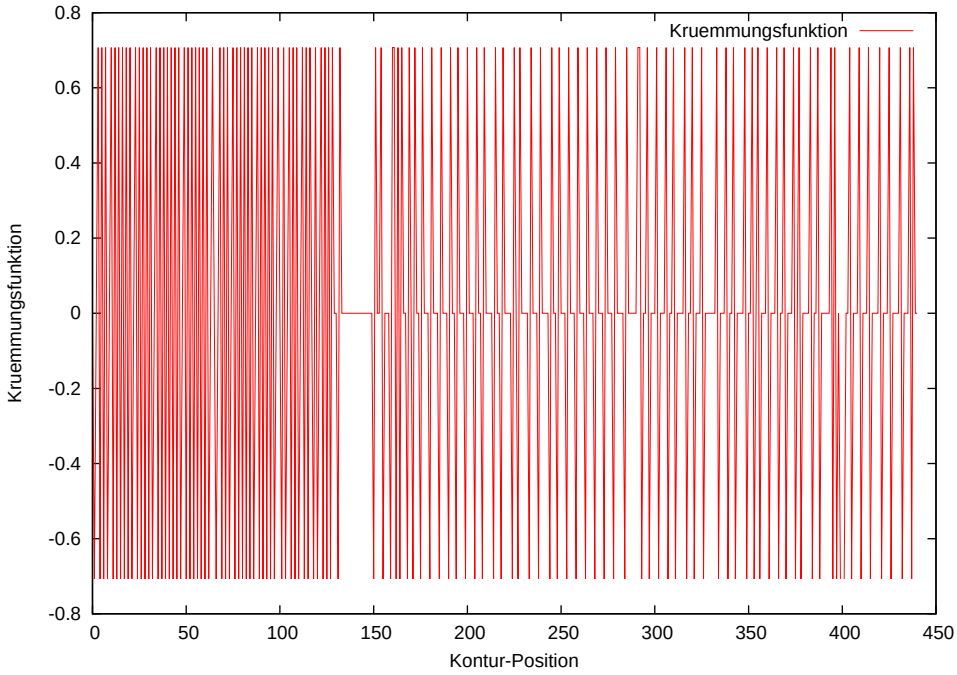


Abbildung 10: Krümmungswerte, berechnet aus den rohen Konturdaten.

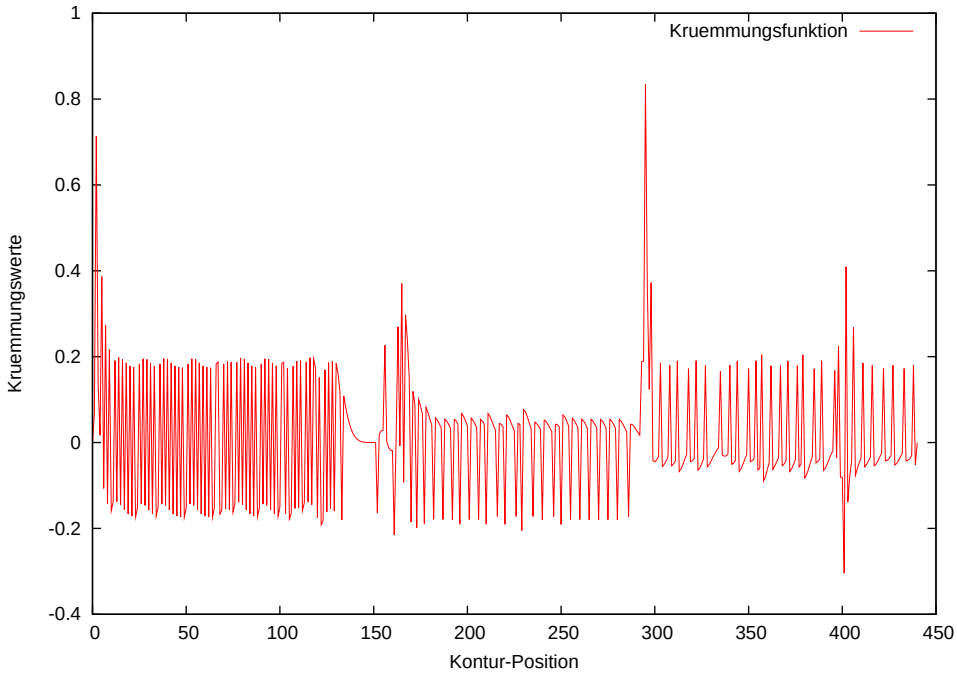


Abbildung 11: Krümmungswerte, berechnet auf den mit Gauß geglätteten Konturdaten für $\sigma = 5$.

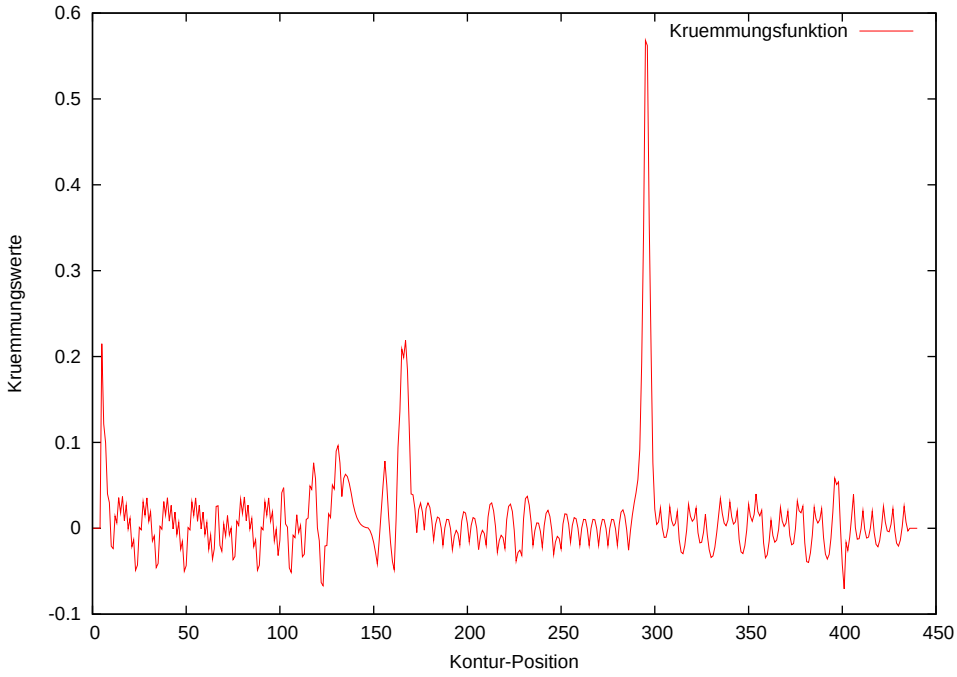


Abbildung 12: Über eine Umgebung gemittelten Krümmungswerte mit $\epsilon = 5$, berechnet auf der Grundlage von Gauß geglätteten Konturdaten mit $\sigma = 5$.

zu berechnen:

$$\begin{aligned} x'(l) &= \frac{1}{R}\sum_{c=1}^{R} x(l+c) - x(l-c) \\ x''(l) &= \frac{1}{R}\sum_{c=1}^{R} x(l+c) + x(l-c) - 2x(l). \end{aligned} \tag{20}$$

Die Verwendung der Gleichung 20 bezweckt einen glatteren Verlauf der Krümmungsfunktion, bei dem ein Eckpunkt mit einem deutlichen lokalen Extremum verbunden werden kann. Die mit der Gleichung 20 ermittelten Krümmungswerte sind in Abbildung 9 (unterste Reihe) zu sehen. Darauf erkennt man die Verbesserungen bei den Ableitungen der beiden Ordnungen, insbesondere bei der zweiten, bei der die Spitzen deutlich hervorstehen und sich von den restlichen Daten beträchtlich unterscheiden. Die Eckpunkte lassen sich in Abbildung 12 am deutlichsten erkennen. Die Mittelwertbildung ermöglicht es, den Einfluss der Fehler, verursacht durch die Berechnung der Ableitungen anhand der diskreten Daten, auf die Krümmungsfunktion zu verringern.

Extraktion der Kontrollpunkte und B-Spline Interpolation

Basierend auf den Ergebnissen, die man durch die Anwendung des oben beschriebenen Verfahrens für die Berechung der Krümmungsfunktion erhält, können die Kontrollpunkte extrahiert werden. Für die Gewinnung der Stützstellen werden zuerst die Krümmungswerte absteigend sortiert. Um zu vermeiden, dass sich mehrere Stützpunkte direkt nebeneinander befinden, legt man nach der Auswahl des nächsten verfügbaren Punktes mit dem größten Krümmungswert eine Umgebung um diesen Punkt fest, in der sich keine Kontrollpunkte mehr befinden dürfen. Diese Umgebung muss ausreichend klein gewählt werden, so dass die zukünftige Interpolation nah genug an der ursprünglichen Funktion liegt. Beispielsweise kann diese Umgebung 1/10 oder 1/20 der Konturlänge betragen. Das Ergebnis ist in Abbildung 13 zu sehen.

Subsampling

Für eine korrekte und effiziente Berechnung der Merkmale müssen die Längen der Konturen auf eine einheitliche Größe normiert werden. Diese Größen sind bei der Durchführung der FFT z.B. die Zweierpotenzen. Um die Geschwindigkeit der Berechnung zu erhöhen, ist es günstig, die Anzahl der Punkte so klein wie möglich zu halten. Gleichzeitig muss darauf geachtet werden, dass der Informationsverlust nicht zu hoch ist, d.h., die Kontur sollte vom menschlichen Auge genauso gut zugeordnet werden könnnen wie vor dem Subsampling.

Das *subsampling*-Verfahren bringt die Anzahl der Pixel auf eine konstante Größe k, indem es anhand der Konturlänge den Faktor m ermittelt und dann diesem Faktor entsprechend jeden m-ten Pixel für die Endkontur auswählt. Ein Beispiel der Originalkontur (links) und der Kontur nach dem Subsampling auf 64 Punkte ist in Graphik 39 (Appendix A) abgebildet. Trotz der geringen Punktezahl lässt sich das Objekt genau so gut zuordnen wie das Original.

Im Falle, dass die Pixel-Anzahl kleiner ist als die gewünschte Anzahl k, werden mit einem Interpolationsverfahren von MATLAB (*interp*) die Sampling-Rate der Kontur erhöht, um die Punkteanzahl auf die erwünschte zu bringen.

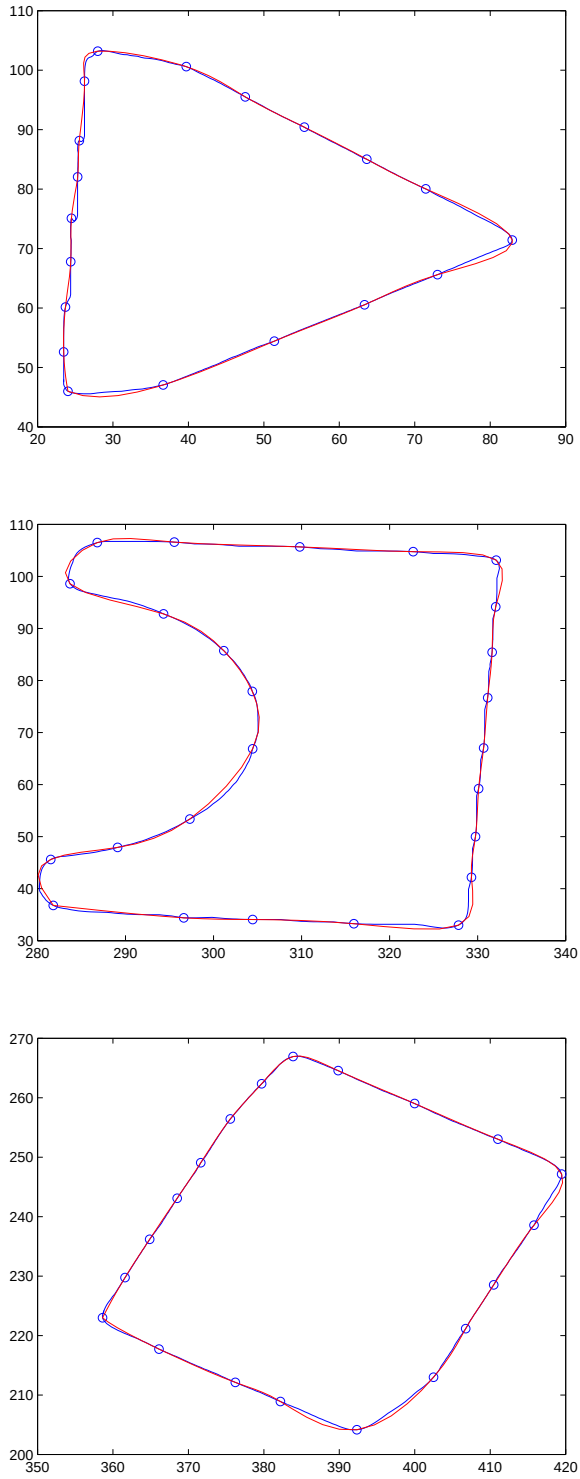


Abbildung 13: Die Kontur mit den ermittelten Kontrollpunkten (blau); durch die Kontrollpunkte gelegte Splines (rot)

4 Merkmalsextraktion

Im Klassifikationsvorgang werden die Konturen durch Merkmale repräsentiert. Dank ihrer Invarianzeigenschaften (Abschnitt 2) können die Fourier-Deskriptoren, Zernike- und pseudo Zernike-Momente im Merkmalsextraktor verwendet werden. Ihr diskriminatives Vermögen wird in den nächsten Abschnitten anhand einiger Abbildungen veranschaulicht. Als Grundlage für die folgenden Versuche dienen 500 Konturen nicht verdeckter Objekte, jeweils 100 für jede Figurklasse. Welches Merkmal im Klassifikator-Ensemble die besten Ergebnisse liefert, wird im Kapitel 6 genauer untersucht.

4.1 Fourier-Deskriptoren

Nach der Normalisierung von Fourier-Deskriptoren (siehe Abschnitt 2.3) erhält man für einen Vektor der Länge L einen reellen Vektor $f = (f_1, \ldots, f_{L-1})$ translations-, skalierungs- und rotationsinvarianter Größen. Der erste Fourier-Koeffizient wird im Merkmalsvektor immer weggelassen, da er an die Translation der Kontur direkt gekoppelt ist. Bekanntlich beschreiben FD niedriger Ordnung die groben Konturverläufe, wobei die FD höherer Ordnung den feineren Struktureigenschaften entsprechen. Beispielsweise erweisen im Falle der fünf Objektklassen (Prisma, Zylinder, Halbzylinder, Brücke und Quader) die normierten Deskriptoren der Ordnung höher als 15 meistens geringfügige Werte nahe Null. Dies wird vermutlich durch das Rauschen in der lokalen Kontursturktur verursacht. Für die Klassifikation vollständiger mit B-Splines modellierter Konturen der fünf Objektklassen sind schon zwei oder drei FD ausreichend.

In Abbildung 14 sind Objektkonturen mit Hilfe von 3 FD f_2, f_4 und f_6 dargestellt. 500 Konturen mit jeweils 100 Konturen pro Figurklasse sind hier abgebildet. Man erkennt, dass jede der fünf Klassen eine Punkwolke bildet. Trotz der perspektivischen Verzerrungen der Konturen, sammeln sich die normierten Fourier-Deskriptoren, unabhängig von der Projektion, in gut trennbare Cluster.

4.2 Zernike und pseudo Zernike-Momente

Vergleichbar gutes diskriminatives Vermögen normierter Zernike- und pseudo Zernike-Momente (siehe Abschnitt 2.5) für die fünf Figurklassen wird durch die Abbildung 15 veranschaulicht. Auf den X-, Y-, und Z-Achsen sind die Momente der zweiten Ordnung

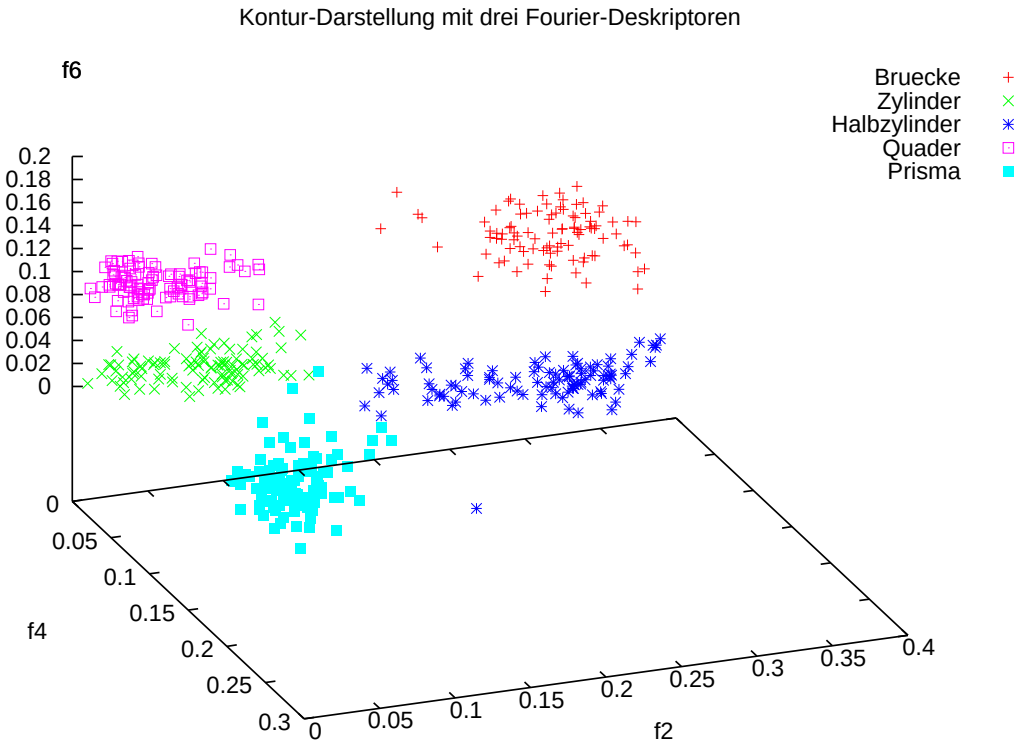


Abbildung 14: Darstellung der 5 Figurklassen mit drei Fourier-Deskriptoren; jede Klasse wird mit 100 Punkten repräsentiert; Brücke (rot), Zylinder (grün), Halbzylinder (blau), Quader (lila), Prisma (hellblau);

M_{20}, der dritten Ordnung M_{33} und M_{31} eingetragen. Die obere Graphik der Abbildung 15 zeigt die Cluster, die durch pseudo Zernike-Momente gebildet werden, die untere stellt die Zernike-Momente dar. Auf den zweidimensionalen Projektionen für FD, ZM und PZM in Abbildungen 14 und 15 kann man beobachten, dass im Vergleich mit FD die normierten Zernike- und pseudo Zernike-Momente deutlich verstreuter liegen. Die jeweiligen Figur-Cluster sind nicht so kompakt gebildet, wie diejenigen mit FD. Das gilt unabhängig von der zweidimensionalen Projektion der Daten in den oben beschriebenen Abbildungen. Ein wahrscheinlicher Grund für diese Verstreutheit ist, dass die ZM und PZM empfindlicher auf die perspektivischen Verzerrungen reagieren als die Fourier-Deskriptoren. Trotzdem bilden sie im dreidimensionalen Raum immer noch gut trennbare Punktwolken.

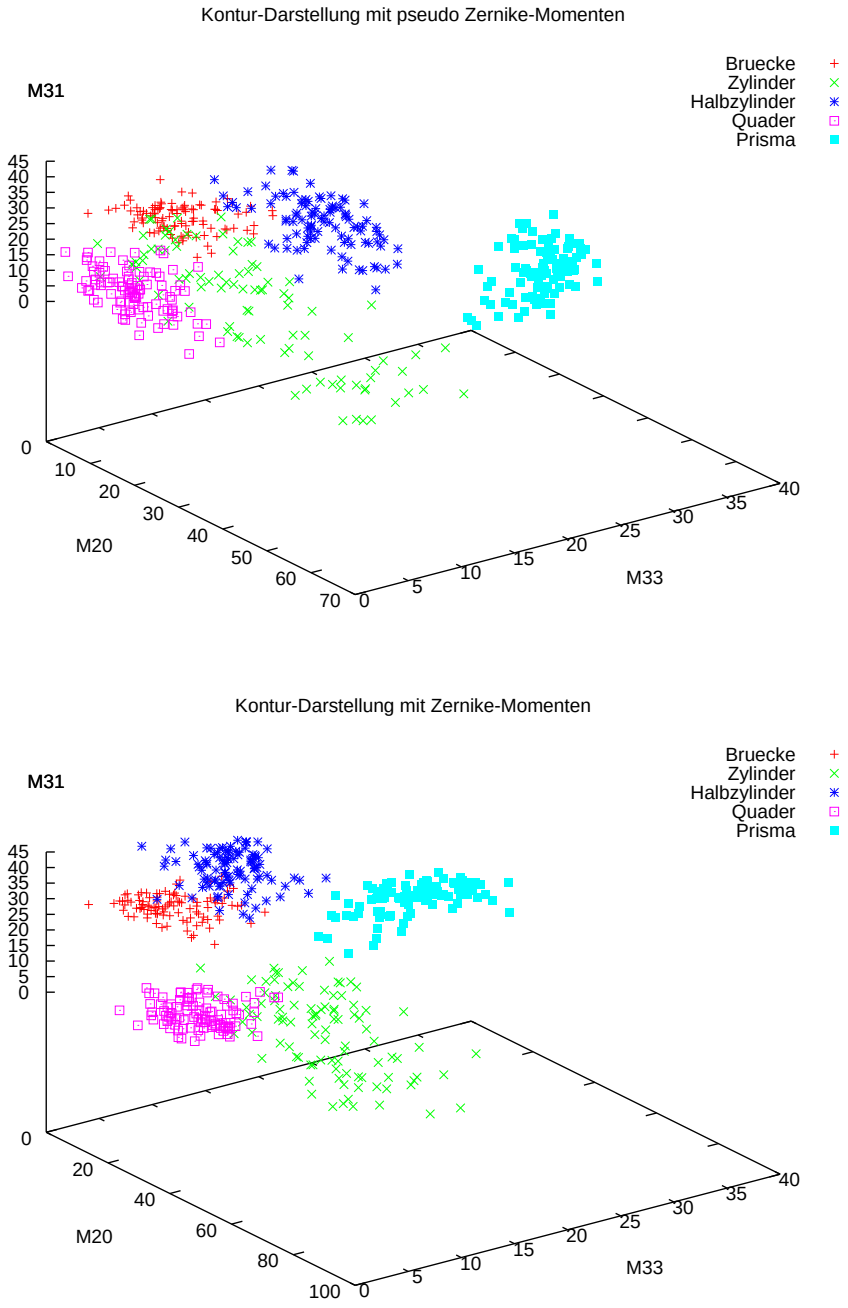


Abbildung 15: Dreidimensionale Darstellung der fünf Figurklassen mit pseudo Zernike-Momenten (oben) und Zernike-Momenten (unten); 100 Punkte repräsentieren die jeweilige Klasse: Brücke (rot), Zylinder (grün), Halbzylinder (blau), Quader (lila), Prisma (hellblau); M_{20}, M_{33}, M_{31};

4.3 Klassifikation vollständiger Konturen

Einige Tests wurden zur Klassifikation vollständiger Konturen der fünf Figurklassen mit einem einzelnen LCC-Klassifikator durchgeführt. Sowohl bei der Verwendung von FD als auch ZM und PZM wurden gute Ergebnisse erzielt. Fehlerraten bei der Klassifikation 250 vollständiger Konturen (50 für die jeweilige Figurklasse) von weniger als 2 Prozent können nach dem Training mit etwa 120 vollständigen Konturen erzielt werden. Der große Nachteil der Zernike- und pseudo Zernike-Momente im Vergleich zu den Fourier-Deskriptoren liegt bei dem erheblichen Zeitaufwand, der für ihre Berechnung notwendig ist.

5 Klassifikator-Ensemble

Ein Klassifikator-Ensemble ist eine Menge von Klassifikatoren, deren individuelle Entscheidungen nach einem Verfahren kombiniert werden (typischerweise gewichtetes oder nichtgewichtetes Voting), um neue Daten zu klassifizieren [12]. Seit den 1980er Jahren werden multiple Klassifikator-Systeme für die Lösung komplexer Aufgaben in Computer-Vision-Anwendungen erfolgreich verwendet. Die schnelle und vielseitige Entwicklung in diesem Bereich wird von T.K. Ho in [9] erläutert und zusammengefasst. Die grundlegenden Konzepte zur Verwendung und zum Aufbau eines Klassifikator-Ensembles werden im folgenden Abschnitt dargelegt.

5.1 Allgemein: Verwendung und Aufbau

Das Ziel einer Klassifikator-Kombination ist die Verbesserung der Klassifikationsgenauigkeit. Allerdings führt die Beteiligung mehrerer Klassifikatoren an einer Entscheidung meistens zum erhöhten Rechenaufwand [23]. Aus diesem Grund ist es wichtig, die Voraussetzungen für einen erfolgreichen Einsatz eines Ensembles zu kennen. In [19] wird für die Verwendung eines Klassifikator-Ensembles eine notwendige und hinreichende Bedingung angegeben. Nach [19] ist der Einsatz eines Klassifikator-Ensembles genau dann vorteilhafter als die Verwendung einer seiner Teile, wenn jeder einzelner Klassifikator *diverse* und *accurate* ist. Klassifikatoren sind divers, wenn sie bei einer neuen Datenmenge verschiedene Klassifikationsfehler machen. Klassifikatoren sind dann akkurat, wenn sie bessere Klassifikationsergebnisse liefern als zufälliges Raten.

Thomas G. Dietterich deutet in [12] auf drei Problembereiche hin, bei denen der Einsatz eines Klassifikator-Ensemble angemessen ist; sie sind: *statistical*, *computational* und *representational* (siehe Abbildung 16). Ein statistisches Problem tritt auf, wenn die Trainingsdaten eine zu kleine Untermenge der Gesamtdaten ausmachen. In diesem Falle können beispielsweise mehrere Klassifikatoren erzeugt werden, die die Daten mit einer ähnlichen Fehlerrate klassifizieren aber unterschiedlich generalisieren. Dann ist es wahrscheinlich, dass ihre Kombination bessere Ergebnisse erzeugt als ein zufällig ausgewählter Klassifikator. Eine Veranschaulichung dieser Gedanken kann man in Abbildung 16 oben links sehen. H bezeichnet in dieser Abbildung den Klassifikator-Raum. Mit blau sind die Klassifikatoren $h_1, \ldots, h_4$ markiert, die ähnlich gute Ergebnisse auf der Trainingmenge liefern. Mit f bezeichnet der Autor den perfekten Klassifikator.

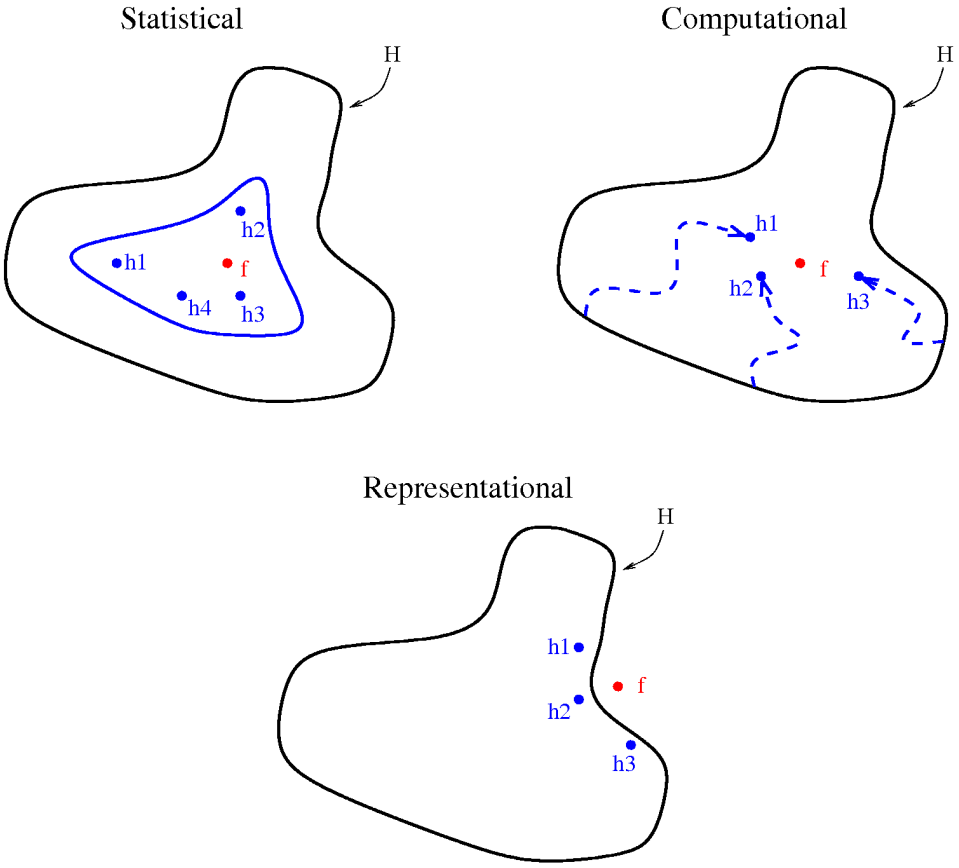


Abbildung 16: Veranschaulichung der drei Gründe für die Verwendung von Klassifikator-Ensemble; die Grafik ist aus [12] entnommen.

Der zweite Grund ein Ensemble zu verwenden ist ein rechnerischer. Mithilfe einer Kombination der Ergebnisse ist es wahrscheinlich eine bessere Approximation des Optimums zu bekommen. Beispielsweise kann ein einzelner Klassifikator bei einer Suche nach dem globalen Minimum in einem lokalen Miminum landen. Bei der Verwendung eines Ensembles verfügt man über mehrere Ausgangspunkte für die Suche und entsprechend auch über mehrere Endergebnisse. In Abbildung 16 rechts wird diese Überlegung veranschaulicht.

Der Repräsentationsgrund wird in der gleichen Abbildung unten dargestellt. Es ist in diesem Falle nicht möglich, f mit nur einem der Klassifikatoren aus der Menge H anzunähern. Mithilfe einer gewichteten Summe der Klassifikatoren läßt sich H erweitern. Beispielsweise lässt sich eine komplexe nicht-lineare Funktion nicht mit einem linearen

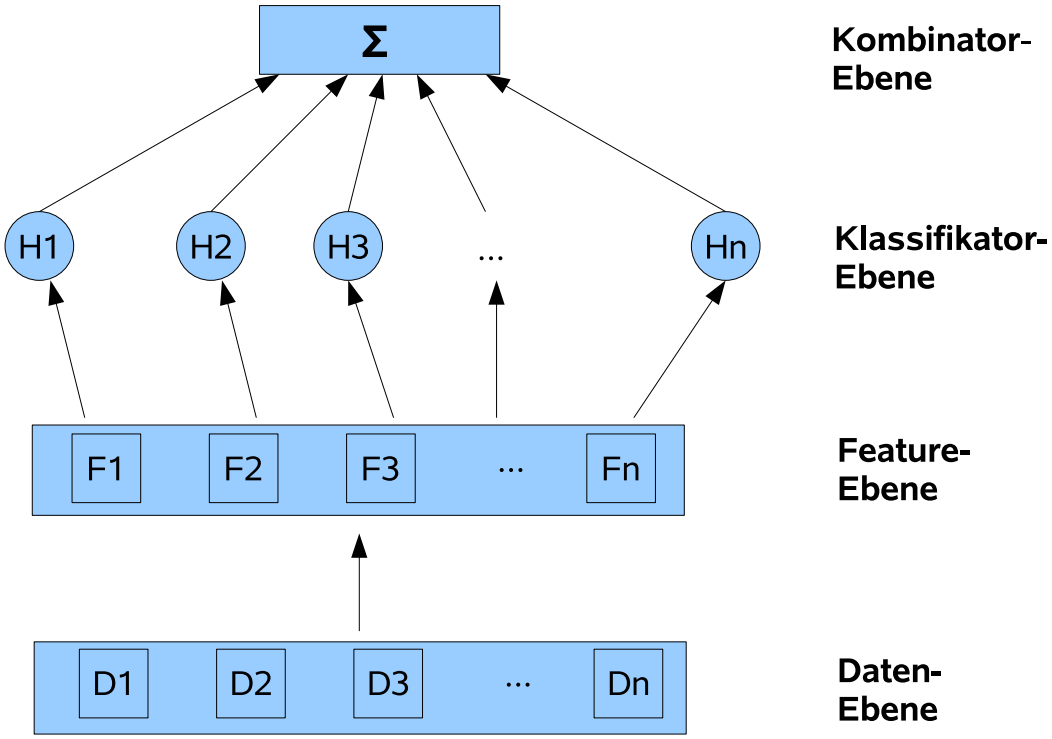


Abbildung 17: Aufbau und Bestandteile eines Klassifikator-Ensembles; Graphikidee entnommen aus [23].

Klassifikator approximieren. Dennoch kann eine Kombination von linearen Klassifikatoren eine gute Approximation für höchst-komplexe nicht lineare Verläufe liefern.

Die Abbildung 17 veranschaulicht die Bestandteile und Aufbauebenen eines Klassifikator-Ensembles: Daten-, Merkmals-, Klassifikator- und Kombinatorebenen. Diesem Aufbau entsprechend teilt die Autorin in [23] die Ensemble-Konzepte in vier Kategorien ein. Auf jeder der oben genannten Ebenen werden verschiedene Konzepte für die Auswahl der

1. Datenmengen,
2. Feature-Mengen,
3. Basis-Klassifikatoren und
4. Kombinationstechniken

eingesetzt. Ganz allgemein kann man auf der untersten Ebene entweder eine Menge verwenden, oder die Daten in mehrere Datenteilmengen $D_1, \ldots, D_n$ unterteilen. Ebenfalls

kann man auf der Merkmalsebene eine einzige oder n verschiedenen Teilmengen $F_1, \ldots, F_n$ des Merkmalsraums verwenden. Die nächste Stufe beschäftigt sich mit der Selektion der Klassifikatoren. In diese Kategorie können alle Methoden aufgenommen werden, die die Basis-Klassifikatoren für den Einsatz im Ensemble auswählen. Die oberste Stufe umfasst alle bestehenden Techniken, die Entscheidungen einzelner Klassifikatoren kombinieren. Dazu wird in der Literatur häufig der Begriff *decision optimization* verwendet. Hierbei geht man davon aus, dass die Menge der Klassifikatoren gegeben ist. Das Ziel besteht darin, bei einer festen Menge an Klassifikatoren eine Kombinationsfunktion zu entwerfen, die die Menge der Einzelentscheidungen so abbildet, dass die Endentscheidung für alle Testdaten optimal ist [9]. Die Bereiche 1 - 3 und die entsprechenden Ebenen in Abbildung 17 werden unter dem Begriff *coverage optimization* zusammengefasst. Es wird angenommen, dass eine Kombinationsfunktion gegeben ist. Die Idee hier ist, eine Menge möglicherweise komplementärer Klassifikatoren so zu erzeugen, dass sie unter der fest gewählten Kombinationsfunktion gute Ergebnisse erzielen. Dabei wird gängig zwischen *classifier fusion* und *classifier selection* unterschieden. Bei *classifier fusion* besitzt jeder Klassifikator im Ensemble das Wissen vom gesamten Merkmalsraum. Im Gegensatz dazu besitzt bei *classifier selection* jedes Mitglied des Ensembles nur die Information über einen bestimmten Teil des Merkmalsraums. Die Autorin [23] merkt an, dass manche Konstruktionsmöglichkeiten nicht von dem Aufbaudiagramm aus Abbildung 17 erfasst werden.

Die existierenden Kombinationskonzepte lassen sich nach der Art der Klassifikator-Ausgabe aufteilen. Im Falle, dass ein Klassifikator nur die Information über den Klassenbezeichner zurückgibt, lassen sich die Kombinationsmethoden wie Mehrheitsentscheidung, gewichtete Mehrheitsentscheidung, Bayes-Kombination, Klassifikator-Kombination mit Singulärwertzerlegung u.a. anwenden. Andererseits können, wenn von einem Klassifikator für ein Datenbeispiel die Information zur Unterstützung einzelner Klassen vorliegt, die Kombinationsmethoden wie weighted average, Fuzzy-Integral, Dempster-Shafer-Kombination u.a. angewendet werden. Weiterhin kann außerdem zwischen class-conscious und class-indifferent Kombinationsfunktionen (siehe Abschnitt 5.5.1) unterschieden werden.

Adaptive Occlusion Classifier ist ein Klassifikator-Ensemble, das innerhalb dieser Arbeit zur Klassifikation der Objektkonturen mit Verdeckung entwickelt wurde. Eine Mischung der oben erläuterten Konzepte wird bei diesem Ensemble verwendet. Die weiteren Abschnitte bieten eine ausführliche Beschreibung der einzelnen Methoden, der Struktur und des Aufbau entsprechend der Abbildung 17 an.

5.2 Datenmenge

Die Ausgangsdatenbasis vom Klassifikator-Ensemble besteht aus einem Pool von Figurkonturen. Diese werden aus den Bildern der nicht verdeckten Objekte extrahiert und anschließend mit dem Verfahren aus dem Abschnitt 3.4 vorverarbeitet. Zwei Methoden, die es ermöglichen, Adaptive Occlusion Classifier mit verdeckten Objekten zu testen, werden im folgenden Abschnitt erläutert.

5.2.1 Verdeckungen

Zur Hauptaufgabe dieser Arbeit gehört die Klassifikation der Konturen teilweise verdeckter Objekte. Es existieren zwei Möglichkeiten solche Konturen zu erhalten: Extraktion aus Bildern verdeckter Objekte oder künstliche Erstellung. Um eine einfache und schnelle Durchführung der Tests zu ermöglichen, werden hier die Kontur-Verdeckungen künstlich generiert. Einige Beispiele sind in Abbildung 18 zu sehen. Die zwei Verfahren für die Herstellung der künstlichen Verdeckungen innerhalb dieser Arbeit approximieren die echten Verdeckungen mit einer linearen Funktion. Das eine Verfahren verwendet dabei die Konturdaten, das andere die Schwarzweißbilder.

Das erste Verfahren wird in der Arbeit *Kontur-Verdeckung* genannt und besteht aus zwei einfachen Schritten. Als Erstes wird ab einer zufällig gewählten Stelle ein Stück der Objektkontur weggelassen. Hierbei wird der entsprechenden Funktion ein Parameter β übergeben. Dieser gibt den Anteil der Konturlänge an, der gelöscht werden soll. In der Realität besitzen die verdeckten Objekte eine lückenlose Kontur, deshalb muss das im ersten Schritt entstandene Konturstück im zweiten Schritt geschlossen werden. In der Implementierung werden die beiden Endpunkte mit einer Geraden verbunden. Somit ist hier die Form der Verdeckung auf eine *straight-line boundary* eingeschränkt, die weiterhin vereinfachend als linear bezeichnet wird. Innerhalb des praktischen Teils der Arbeit werden Versuche für konstant oder randomisiert gewählte βs durchgeführt. Im Falle einer konstanten Verdeckung bleibt für alle Testbeispiele der Parameter β konstant und wird innerhalb dieses Versuchs mit β_{const} bezeichnet. Die andere Möglichkeit ist, innerhalb der Sample-Menge eine uniform randomisierte Verdeckung mit einer oberen Schranke $\beta_{max} \in [0, 1)$ zu erzeugen. Hierfür wird für jedes Sample ein neuer β mit uniformer Wahrscheinlichkeit aus dem Intervall $[0, \beta_{max}]$ gewählt.

Die oben beschriebene Methode zur Generierung künstlicher Verdeckungen hat eine

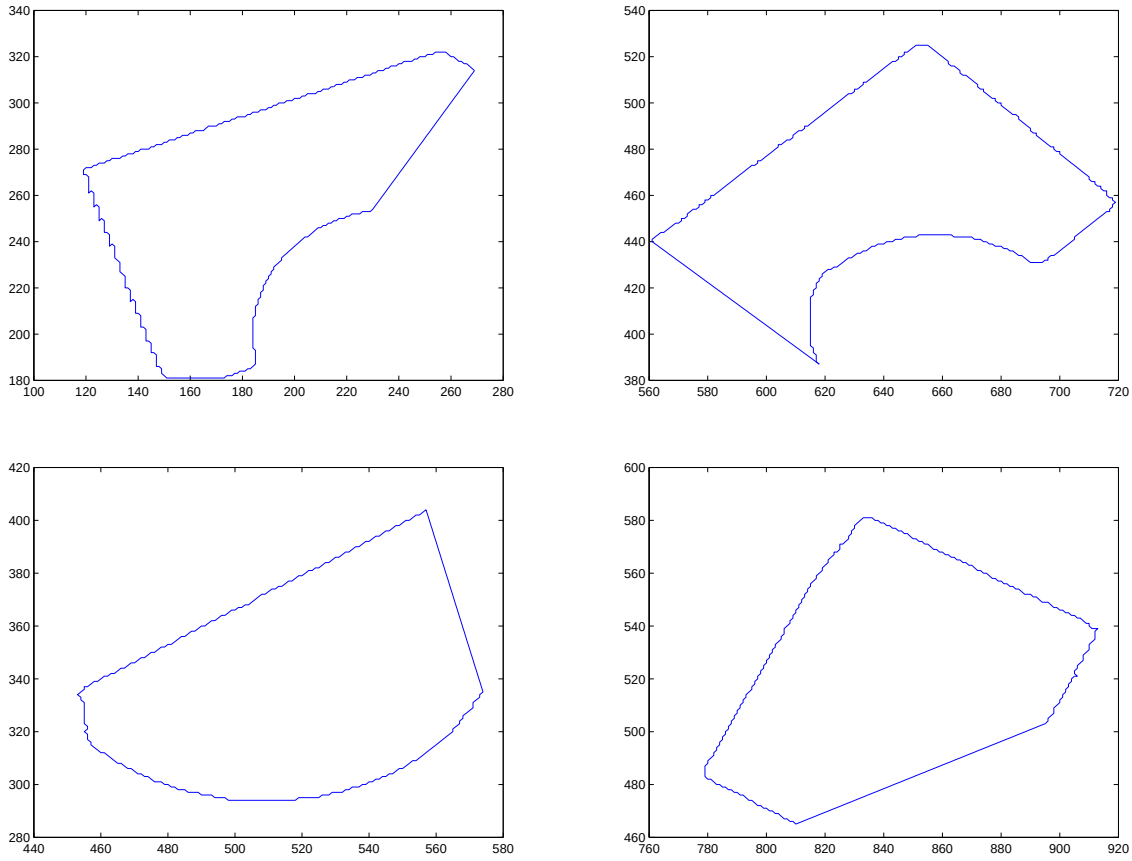


Abbildung 18: Figurkonturen mit synthetisch generierten Verdeckungen; Brücke, Halbzylinder und Quader.

Unzulänglichkeit. Diese besteht darin, dass im Falle nicht konvexer Objektformen Konturen erzeugt werden können, die in der Realität nicht vorkommen. Zwei typische Fehler bei der Verdeckungsproduktion für Brückenkonturen liegen in Abbildung 19 vor.

Das zweite Verfahren generiert die Verdeckungen im Gegensatz zum ersten nicht aus den Konturen, sondern aufbauend auf Objektflächen im Schwarzweißbild. Man erreicht dadurch, genau einen vorgegebenen Anteil der Objektfläche zu eliminieren, dem entsprechend wird es *Flächen-Verdeckung* genannt. Der Rechenaufwand dieses Verfahrens ist höher, dafür besitzt es nicht die Unzulänglichkeit des ersten Verfahrens. Außerdem sind die resultierenden Verdeckungen realitätsnah. Als Erstes wird mit einer uniformen Verteilung aus dem Intervall $[0, 2\pi]$ ein Winkel ϕ ausgewählt. Aus ϕ konstruiert man den folgenden Vektor der

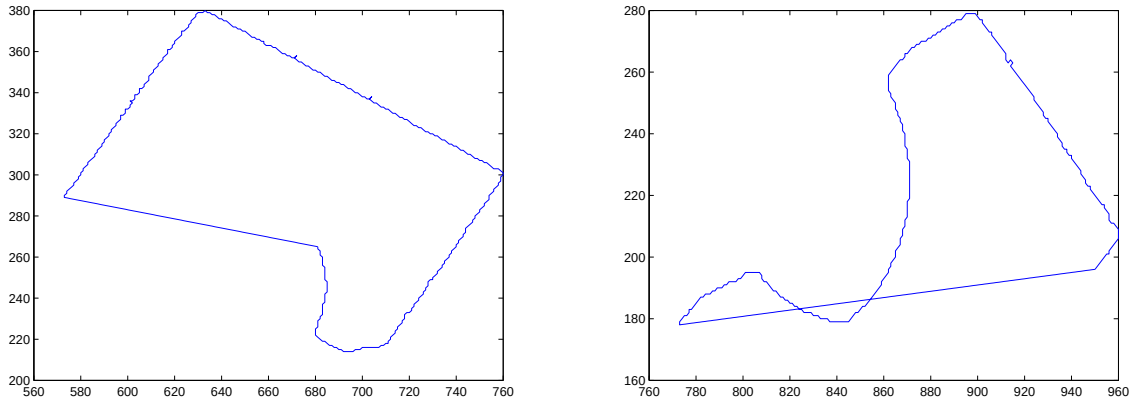


Abbildung 19: Defekte bei der künstlichen Erzeugung der Verdeckungen von Brückenkonturen.

Länge eins:

$$e = (e_x, e_y) = (\sin(\phi), \cos(\phi)).$$

Als Nächstes betrachtet man ein Binärbild, wobei die Intensitätswerte der Figur-Pixel eins betragen (siehe Abbildung 20 - links). Sei $B \in \mathbb{N}$ die Breite des Bildes und $H \in \mathbb{N}$ ihre Höhe. Dann ist die Menge aller Figur-Pixel des Bildes folgendermaßen definiert[1]:

$$W = \{(x, y) \in \underline{B\rfloor} \times \underline{H\rfloor} | I_{xy} = 1\}.$$

Sei f eine Funktion, die durch das folgende Skalarprodukt definiert ist:

$$f : W \to \mathbb{R}, \quad (x, y) \mapsto \langle (x, y) | e \rangle \, .$$

Der Gradient von f ist eine konstante Funktion:

$$\text{grad}(f) : W \to \mathbb{R}^2, \quad (x, y) \mapsto (e_x, e_y). \tag{21}$$

Um den erwünschten Anteil $a \in [0, 1]$ der Originalfläche zu verdecken, wird als Erstes die Menge $S = \{f(x, y) | (x, y) \in W\}$ gebildet. Die Graphik 20 (Mitte) illustriert das Prinzip vom resultierenden Skalarfeld S, seinen Bezug zum Vektor e und veranschaulicht die Gleichung 21. Im Beispiel besteht das Dreieck aus 22 Pixel. Wenn a den erwünschten Verdeckungsanteil angibt, dann ist $n = a|W|$ die Anzahl der zu eliminierenden Figur-Pixel.

[1]Hier und in den weiteren Abschnitten ist $\underline{N\rfloor}$ für ein beliebiges $N \in \mathbb{N}$ definiert durch $\underline{N\rfloor} := \{1, \ldots, N\}$.

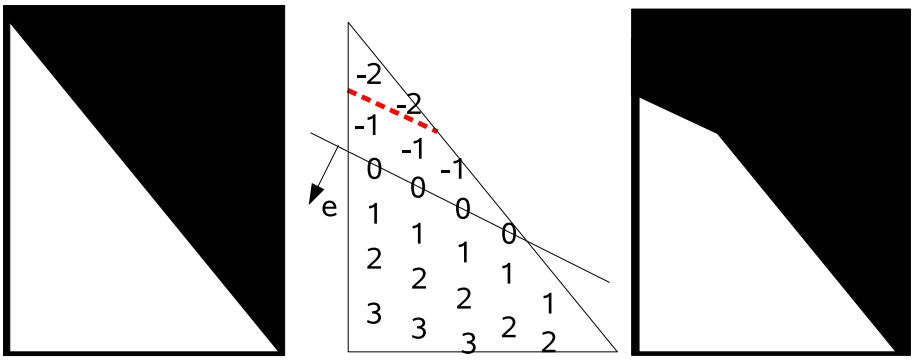


Abbildung 20: Ein Beispiel der Funktionsweise vom zweiten Verdeckungsverfahren für $n = 2$. Links: ein Binärbild von einem Dreieck; Mitte: Skalarfeld berechnet für die Figur-Pixel; Rechts: generierte Verdeckung vom Binärbild.

In der Implementierung wird die Menge S aufsteigend geordnetet. Anschließend werden die Intensitätswerte der Pixel, die den ersten n-Funktionswerten der geordneten Menge entsprechen, auf null gesetzt. Damit erzielt man eine lineare Verdeckung vom Anteil genau gleich a der Originalfläche. Das MATLAB-Verfahren zur Konturextraktion wird an dieser Stelle, wie im Abschnitt 3.4.1 beschrieben, auf dem resultierenden Schwarzweißbild (siehe Abbildung 20 - rechts) durchgeführt. Im experimentellen Teil werden analog zur ersten Verdeckungsmethode konstante und randomisierte Verdeckungen verwendet. Dabei bezeichnet man die obere Schranke für eine randomisierte Verdeckung der Sample-Menge mit γ_{max} und einen konstanten Verdeckungsparameter für alle Samples mit γ_{const}.

Um die beiden Methoden vergleichen zu können, werden nach der Verdeckung resultierende Flächenanteile und Konturanteile betrachtet. Die Abbildung 21 stellt die Umrechnung von Kontur- in Flächen-Verdeckung dar. Hier sind die durchschnittlichen Anteile der verdeckten Flächen dargestellt, die sich aus dem Verfahren Kontur-Verdeckung ergeben. Auf der X-Achse ist der Parameter β_{const} eingetragen. Eine Verdeckung β_{const} bedeutet die Anwendung vom Verfahren Kontur-Verdeckung zur Generierung einer konstanten Verdeckung innerhalb der Sample-Menge. Hierbei wird bei allen Samples aus der Testmenge ein Anteil β_{const} der Kontur ab einer zufälligen Stelle weggelassen, um die entstehende Lücke mit einer Geraden zu schließen. An der Y-Achse sind die für 1000 Samples gemittelten Werte der entstandenen Flächen-Verdeckung als Anteil der vollständigen Fläche eingetragen. Diese Graphik zeigt die Abhängigkeit der γ_{const} vom Parameter β_{const}. Die Flächen

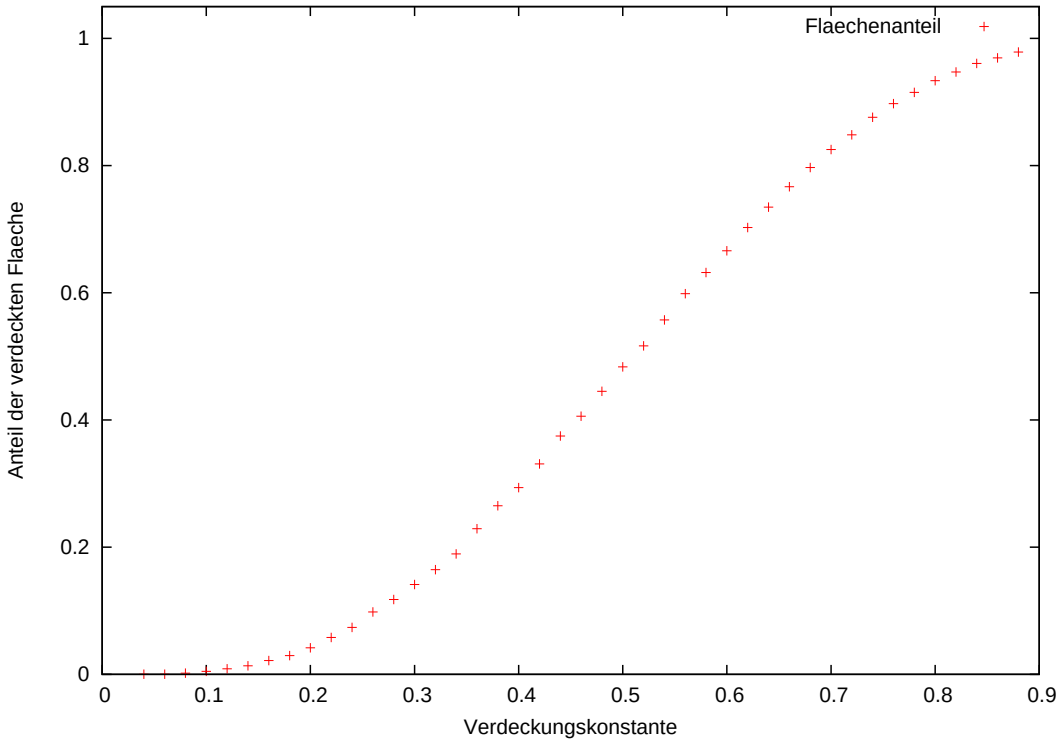


Abbildung 21: Flächen-Verdeckung resultierend aus dem Verfahren Kontur-Verdeckung. Der Ergebniswert für eine gegebene Verdeckungskonstante β entspricht dem Mittelwert über 1000 Samples, jeweils 200 Samples pro Figurklasse.

werden mit dem Flutfüllung-Algorithmus berechnet. Wie man in Graphik 21 sehen kann, wächst β_{const} bis ca. 0.5 schneller als γ_{const}, danach gilt das Umgekehrte.

In Abbildung 22 kann man die aus den beiden Verfahren resultierenden Anteile der Konturlängen vergleichen. Auf der X-Achse ist hier die Verdeckungskonstante eingetragen. Im ersten Verfahren wird dieser Wert für β_{const} eingesetzt, im zweiten für γ_{const}. Die Y-Achse stellt die gemittelten Anteile der vollständigen Konturen dar, die sich für die entsprechenden Werte der Verdeckungskonstanten ergeben. In dieser Abbildung kann man leicht erkennen, dass bis $\beta_{const} = \gamma_{const} \approx 0.5$ die aus dem ersten Verfahren resultierenden Werte über denen von dem zweiten liegen. Bis zu diesem Punkt geht also durch das zweite Verfahren mehr Konturinformation verloren. Ab $\beta_{const} = \gamma_{const} \approx 0.5$ gilt das Umgekehrte.

Einem β_{max} entsprechendes γ_{max} für den Vergleich der Versuchsergebnisse mit einer randomisierten Flächen-Verdeckung und einer randomisierten Kontur-Verdeckung kann mit Hilfe der Daten aus Abbildung 21 ermittelt werden. Bezeichne die Funktion aus dieser

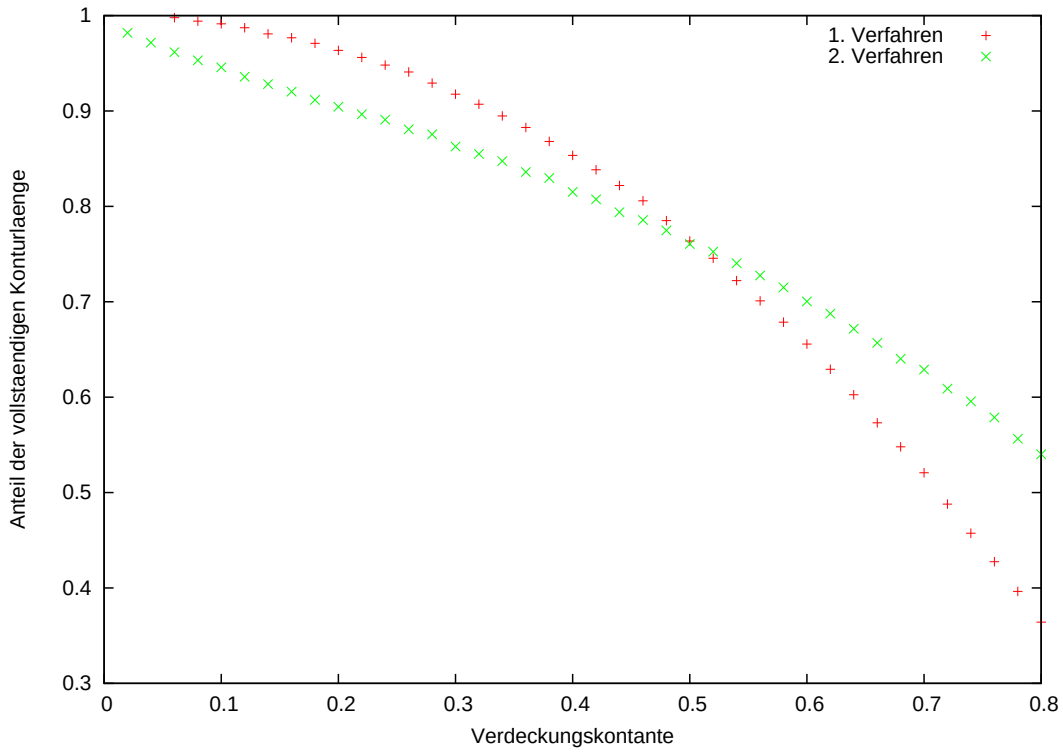


Abbildung 22: Vergleich der Anteile der Konturlängen mit dem ersten und zweiten Verdeckungsverfahren; Mittelwert berechnet über 1000 Testkonturen, jeweils 200 Samples pro Figurklasse.

Abbildung mit v und es gilt:

$$v : [0,1] \to [0,1], \quad \beta_{const} \mapsto \gamma_{const}.$$

Die experimentell ermittelte Funktion v gibt annähernd an, welche Flächen-Verdeckung γ_{const} einer gegebenen konstanten Kontur-Verdeckung β_{const} entspricht. Sei X eine uniformverteilte Zufallsvariable, definiert für ein gegebenes β_{max} durch:

$$X : \Omega \to [0, \beta_{max}], \quad \omega \mapsto \beta$$

und Y definiert als eine Hintereinanderausführung von X und v:

$$Y = v \circ X.$$

Die Verteilung von Y ist durch die Funktion v definiert. Es lässt sich somit der Erwartungswert $E(Y)$ berechnen:

$$\mathrm{E}(Y) = \frac{1}{N} \sum_{i=1}^{N} v(\beta_i), \tag{22}$$

wobei $\beta_i \in [0, \beta_{max}]$ und β_is uniform im Intervall verteilt sind.

Aus dem gesuchten Intervall $[0, \gamma_{max}]$ sollen γs ebenfalls mit einer uniformen Verteilung ausgewählt werden. Aufbauend darauf lässt sich mit Gleichung 22 für einen β_{max} die entsprechende obere Schranke γ_{max} ermitteln:

$$\gamma_{max} = 2\mathrm{E}(Y). \tag{23}$$

Im Abschnitt 6 werden die Klassifikationsergebnisse für die beiden Methoden Kontur-Verdeckung und Flächen-Verdeckung präsentiert und verglichen.

5.2.2 Hierarchische Fragmentierung

Im Abschnitt 3.4.3 wird ein Verfahren zur Extraktion der Kontrollpunkte vorgestellt. Die daraus resultierende Punktmenge, die anfangs zur B-Spline Modellierung verwendet wird, enthält die besonders bedeutsame Teilmenge aller Eckpunkte (siehe Abbildung 13). Betrachtet man die Konturstücke zwischen jeweils drei benachbarten Kontrollpunkten, so erhält man die Menge *elementarer klassen-spezifischer Konturfragmente*. Bei der Quader-Klasse sind es beispielsweise verschiedene Ecken und gerade Linien, bei der Brücken-Klasse sind es Ecken, gerade Linien und auch Rundungen. Im Weiterem werden solche Konturabschnitte vereinfachend *elementare Fragmente* genannt. Allgemein werden elementare Fragmente oder eine Konstellation mehrerer benachbarter elementarer Fragmente mit dem kumulativen Begriff *Fragment* bezeichnet. Fragmente mit der Ausnahme der geraden Linien werden in dieser Arbeit *charakteristisch* genannt.

Für die Klassifikation ist ausschlaggebend, dass im Falle einer Verdeckung die charakteristischen Fragmente verloren gehen können. Die Erkennung der Formen teilweise verdeckter Objekte erfordert daher eine Methode, die lokal die übriggebliebenen, elementaren charakteristischen Fragmente und ihre Konstellationen analysieren und einer Klasse zuordnen kann.

Beruhend auf dem Segmentierungsverfahren aus dem Abschnitt 3.4.3 wurde von H. Prehn das Verfahren *hierarchische Fragmentierung* entworfen und implementiert, das verschiedene Fragmente-Mengen gemäß ihrer strukturellen Komplexität aus einer Kontur erstellt. Hierarchische Fragmentierung ist der Grundstein der Datenverarbeitung, der die Klassifikation verdeckter Objekte ermöglicht. Für eine gegebene Kontur und die Menge dazugehöriger Kontrollpunkte gibt das Verfahren im ersten Schritt die Menge elementarer Fragmente zurück. Im i-ten Schritt vereinigt sie nacheinander die benachbarten Fragmente,

die im $(i-1)$-ten Schritt erhalten wurden und gibt sie ebenfalls zurück. Es wird so lange iteriert, bis man bei der vollständigen Kontur angelangt ist. In Abbildung 23 ist ein Beispiel der hierarchischen Fragmentierung einer Brückenkontur zu sehen. Diese Abbildung stellt die aus dem Verfahren resultierenden fünf Segmentierungsebenen dar. Die erste Ebene enthält die elementaren Kontursegmente. Diese werden in den nachfolgenden Ebenen zu immer größeren und strukturell-komplexeren Stücken zusammengesetzt. Die letzte Ebene besteht aus vollständigen Konturen.

Bezeichnet man die Iterationen-Anzahl des Verfahrens einer gegebenen geschlossenen Kontur mit n, so entstehen nach der hierarchischen Fragmentierung n Fragmente-Mengen, die zur Berechnung der Merkmale an den Merkmalsextraktor getrennt weitergegeben werden.

5.3 Featuremenge

Im obigen Abschnitt ist dargestellt worden, in welche Fragmentmenge eine Kontur mit Hilfe der hierarchischen Fragmentierung überführt wird. Zusammengefasst wird eine gegebene Kontur vor der Merkmalsberechnung erst segmentiert und mit B-Splines modelliert. Dieser Vorgang bezweckt die Beseitigung des Rauschens und der Datenreduktion. Hierarchische Fragmentierung mit der Menge der B-Spline-Kontrollpunkte ergibt mehrere Ebenen von Fragmenten strukturell *hierarchischer* Komplexität. Ausgehend von der ersten Ebene enthalten die Fragment-Konstellationen der nachfolgenden Ebenen eine monoton-wachsende Anzahl elementarer charakteristischer Fragmente. Vor der Merkmalsberechnung müssen alle lückenhaften Fragmente analog zum Abschnitt 5.2.1 mit einer Geraden geschlossenen und auf eine richtige Pixel-Anzahl gebracht werden (siehe Abschnitt 3.4.3). Ein Merkmalsvektor bestehend aus FD, ZM oder PZM kann dann abhängig von der Angabe der Vektorlänge ermittelt werden. Mit der hierarchischen Fragmentierung werden die Daten für Klassifikator-Ensemble-Training und -Test aus den Training- und Testkonturen erzeugt.

Im perfekten Fall würde sich nach dem Klassifikator-Training ein Modell ergeben, das den Merkmalsraum teilweise verdeckter Objektkonturen vollständig erfasst und dadurch eine Klassenzuordnung beliebig verdeckter Testobjekte ermöglicht. Die Mächtigkeit der Menge aller Verdeckungen ist enorm hoch, deshalb bleibt man bei der Auswahl der Trainingsdaten auf eine approximative Lösung angewiesen. Innerhalb dieser Arbeit werden deshalb im Training die Verdeckungsstufen einer vollständigen Kontur betrachtet, die

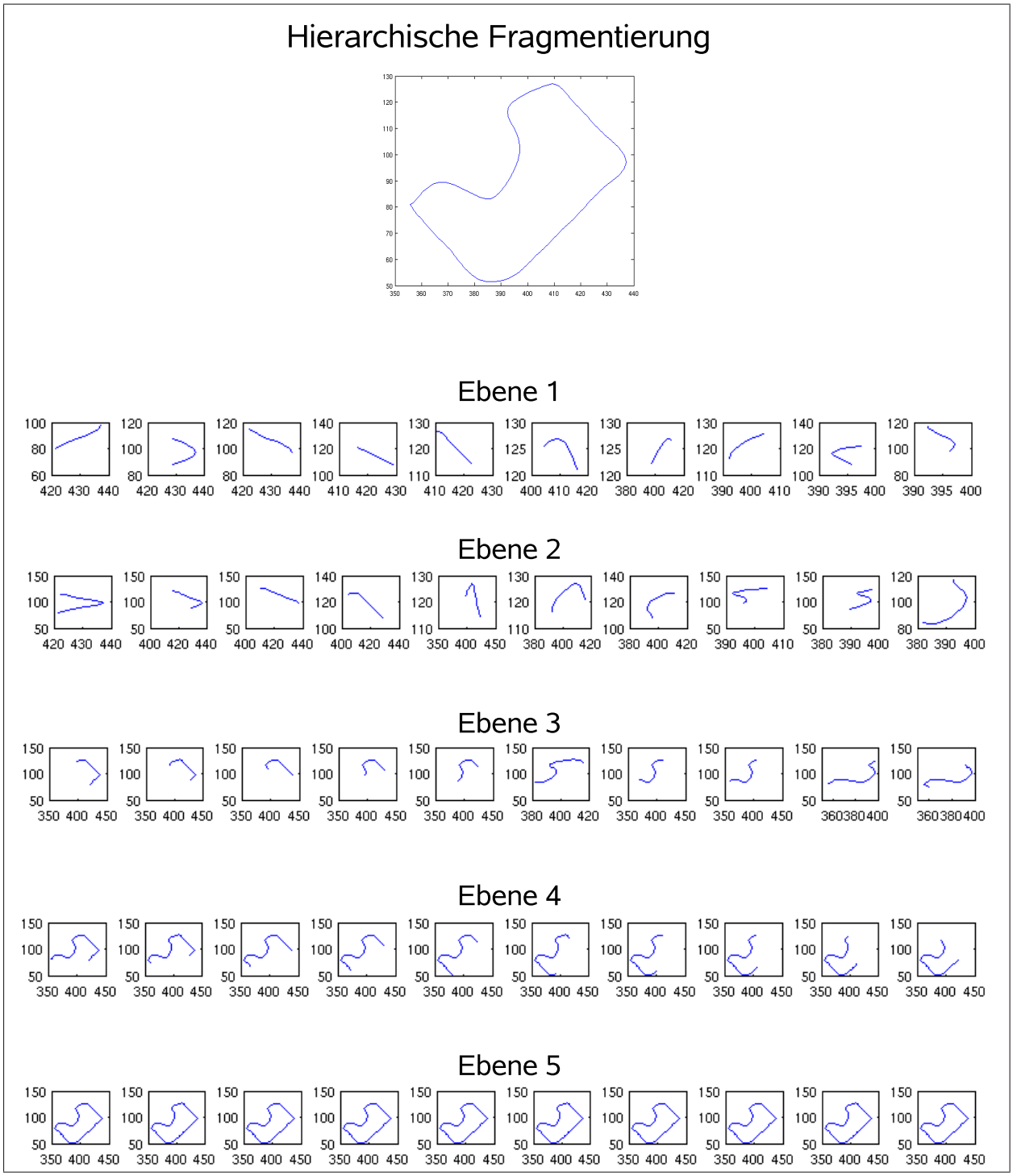


Abbildung 23: Ein Beispiel der hierarchischen Fragmentierung einer Brückenkontur: Ebenen eins bis fünf.

durch die entsprechenden hierarchischen Ebenen nach der Fragmentierung definiert sind. Im Testvorgang werden ebenfalls die hierarchischen Ebenen eines Samples untersucht. Damit erzielt man eine lokale Analyse der verdeckten Kontur, wobei sowohl die elementaren charakteristischen Fragmente als auch ihre Konstellationen auf die Zugehörigkeit zu einer Klassen geprüft werden.

Jedes Mitglied des Klassifikator-Ensembles spezialisiert sich auf seine eigene Verdeckungsstufe bzw. hierarchischen Ebene. Die einzelnen Ensemble-Mitglieder arbeiten mit den Feature-Mengen, die aus den Fragmenten der entsprechenden Ebenen berechnet werden. Innerhalb des Ensembles werden die individuellen Mitgliederentscheidungen mit Hilfe eines gewichteten Durchschnitts zusammengefügt (Abschnitt 5.5).

5.4 Basis-Klassifikatoren

Alle verwendeten Klassifikatoren des Ensembles sind *Local Credibility Criterion (LCC)* [28] Klassifikatoren. Sei eine Datenmenge mit den dazugehörigen Klassenbezeichnern gegeben durch $(x_t, c_t)_{t \in \underline{N}\rfloor}$, wobei $x_i \in \mathbb{R}^d$, $d \in \mathbb{N}$, und $c_t \in \underline{K}\rfloor$. Sei K die Anzahl der Klassen. Ein LCC-Klassifikator besteht aus mehreren Modellen im Form von Hypersphären, deren *credibility* γ durch das Verhältnis der Anzahl korrekter Antworten R_c zur Anzahl der Gesamtantworten R_t definiert ist:

$$\gamma = R_c / R_t.$$

Sei M_k die Anzahl der Modelle, die jede Klasse $k \in \underline{K}\rfloor$ repräsentieren und $i \in \{1, \ldots, M_k\}$. Jedes Modell m_{ki} ist definiert durch einen Schwerpunkt $c_{ik} \in \mathbb{R}^d$ und eine invertierbare Gewichtematrix W_{ki}. Sowohl die Modelle selbst als auch ihre Anzahl ist dynamisch. Der Anpassungsprozess wird über eine Menge von Schwellwerten (siehe [28]) geregelt.

Die Zugehörigkeit eines Samples x zum Modell m_{ki} wird durch die folgende Funktion definiert:

$$s(m_{ki}, x) = 1 - (x - c_{ki})^T W_{ki}^{-1} (x - c_{ki}). \tag{24}$$

Das Sample x gehört zum Modell m_{ki}, wenn der in Gleichung 24 berechnete Wert positiv ist. Die Antwort des Modells m_{ki} auf das Sample x ist gegeben durch:

$$d(m_{ki}, x) = s(m_{ki}, x)\gamma_{ki}.$$

Die Antwort der Klasse k wird folgendermaßen berechnet:

$$d_k(x) = \sum_{\substack{i=1 \\ d(m_{ki},x)>0}}^{M_k} d(m_{ki}, x). \tag{25}$$

Der Klassenbezeichner wird entsprechend dem Maximum der $d_k(x)$s zugeordnet:

$$\arg_{k \in K} \max \; d_k(x). \tag{26}$$

Im Ensemble werden genau soviele Klassifikatoren verwendet, wie die resultierende Anzahl der Ebenen nach der hierarchischen Fragmentierung der Daten ist. Jeder Klassifikator wird mit den Features trainiert und getestet, die seiner Ebene ensprechen.

5.5 Kombinationstechniken

In dieser Arbeit werden zwei verschiedene Kombinationsverfahren verwendet: Multi-Class AdaBoost [40] und gewichteter Durchschnitt, wobei für letzteres die Gewichte mit Hilfe eines linearen Optimierungsverfahren ermittelt werden.

Sei $F \subset \mathbb{R}^n$ ein Merkmalsraum und $C := \{1, \ldots, K\} \subset \mathbb{N}$ die Menge der Klassenbezeichner. Sei $D := \{D_1, \ldots, D_L\}$ das Klassifikator-Ensemble, wobei ein Klassifikator D_i, $i \in \underline{L}$ sich mit der folgenden Abbildung beschreiben lässt:

$$D_i : F \to [0,1]^K, \quad x \mapsto (d_{i1}(x) \ldots d_{iK}(x)).$$

Somit ordnet der Klassifikator D_i dem Merkmalsvektor x einen Vektor zu, der die einzelnen Klassenantworten des Klassifikators enthält. Das *Decision profile* $DP(x)$ eines Klassifikator-Ensembles D zum Merkmalsvektor x ist eine Matrix der Form:

$$DP(x) := \begin{pmatrix} d_{11}(x) \ldots d_{1j}(x) \ldots d_{1K}(x) \\ d_{21}(x) \ldots d_{2j}(x) \ldots d_{2K}(x) \\ \ldots\ldots\ldots\ldots\ldots \\ d_{L1}(x) \ldots d_{Lj}(x) \ldots d_{LK}(x) \end{pmatrix}$$

Die i-te Zeile der Matrix entspricht der Antwort des i-ten Klassifikators auf den Merkmalsvektor x. Die j-te Spalte entspricht der Unterstützung der j-ten Klasse durch die einzelnen Klassifikatoren $D_1, \ldots, D_L$. Die einzelnen Klassifikatoren des Ensembles besitzen nicht die gleiche Klassifikationsgenauigkeit. Es ist daher naheliegend, die Klassifikator-Entscheidungen entsprechend ihrer Leistung bei der Endentscheidung zu gewichten.

Im Weiteren werden die Kombinationstechniken beschrieben, die im Ensemble eingesetzt werden.

5.5.1 Gewichteter Durchschnitt

Die Antwort des Klassifikator-Ensembles D auf ein Sample x kann durch die Bildung des gewichteten Durchschnitts ermittelt werden. Für ein $j \in K$ existieren laut [23] die folgenden drei Varianten:

1. L**-Gewichte**:
$$\mu_j(x) = \sum_{i=1}^{L} w_i d_{ij}(x) \tag{27}$$

2. $(K \times L)$**-Gewichte**:
$$\mu_j(x) = \sum_{i=1}^{L} w_{ij} d_{ij}(x) \tag{28}$$

3. $(K \times K \times L)$**-Gewichte**:
$$\mu_j(x) = \sum_{i=1}^{L} \sum_{k=1}^{K} w_{ikj} d_{ik}(x) \tag{29}$$

In Gleichung 27 wird der Durchschnitt über die $j$te Spalte vom $DP(x)$ gebildet. Diese Kombinationsfunktion verwendet die Gewichte-Menge $\{w_i | i \in \underline{L}\rfloor\}$, wobei für einen Klassifikator D_i eine einzige Gewichtung w_i für alle seine Klassenantworten zur Verfügung steht.

Bei Gleichung 28 wird ebenfalls die $j$te Spalte vom $DP(x)$ in die Berechung des Durchschnitts einbezogen. Der Unterschied zum ersten Verfahren besteht darin, dass die Gewichtsmenge $\{w_{ij} | i \in \underline{L}\rfloor,\ j \in \underline{K}\rfloor\}$ klassenspezifisch ist. Die Kombinationsfunktion verwendet für die Antworten des Klassifikators D_i die Gewichte w_{ij}, die abhängig von der vorgegebenen Klasse j gewählt werden. Die Methode wird entsprechend *class-conscious* genannt.

Im Gegensatz zu den Gleichungen 27 und 28, setzt man bei der dritten Methode (Gleichung 29) alle Spalten des Entscheidungsprofils $DP(x)$ ein. Die Gewichtsmenge $\{w_{ikj} | i \in \underline{L}\rfloor,\ k \in \underline{K}\rfloor,\ j \in \underline{K}\rfloor\}$ wird für die Berechnungen verwendet, wobei alle für das Sample x vorhandenen Antworten zum gewichteten Durchschnitt beitragen. Diese Kombinationstechnik nennt man dementsprechend *class-indifferent*.

In dieser Arbeit wurde die zweite Methode wegen ihren guten experimentellen Ergebnisse angewendet.

5.5.2 AdaBoost und SAMME

In [14] wird von den Autoren die folgende Beschreibung des Boosting-Konzeptes angegeben:

> *Boosting refers to this general problem of producing a very accurate prediction rule by combining rough and moderately inaccurate rules-of-thumb.*

Der AdaBoost (Adaptive Boosting) Algorithmus stammt aus der Arbeit von Yoav Freund und Robert Schapire [15] und ist eins der bekanntesten *boosting* Verfahren. Das Ziel dieses Algorithmus ist, die Klassifikationsgenauigkeit mit Hilfe der gleichzeitigen Anwendung mehrerer komplementärer Klassifikatoren zu verbessern. AdaBoost ist ein überwachtes Lernverfahren. Es generiert die einzelnen Klassifikatoren sequentiell, dabei trainiert es den nachfolgenden Klassifikator mit der falsch klassifizierten Untermenge der Trainingsdaten vorheringer Klassifikatoren. Dieser Prozess wird mit Hilfe einer *adaptiven* Gewichteverteilung koordiniert, die in jedem Schritt neu berechnet wird. Die Gewichteverteilung wird für die Trainingsdaten anhand der vorherigen Klassifikator-Fehler ermittelt und in das Training miteinbezogen.

AdaBoost verlangt von jedem einzelnen Klassfikator eine Fehlerrate kleiner als 0.5. Ist die Fehlerrate größer, so kann AdaBoost scheitern (siehe Beispiel in [40]). Eine vorteilhafte Erweiterung *SAMME*[2] für Multi-Class Probleme wurde in [40] vorgeschlagen. Bei *SAMME* dürfen die einzelnen Klassifikatoren *schwach* sein, d.h. eine Fehlerrate besitzen, die niedriger ist als beim zufälligen Raten. Die erweiterte Version vom Multi-Class AdaBoost Algorithmus, entnommen aus [40], sieht man in Abbildung 24. Der Parameter M bestimmt die erwünschte Anzahl der Trainingvorgänge vom schwachen Klassifikator $D^{(0)}$. In jedem Iterationsschritt wird nach dem Training mit Gleichung 30 die Fehlerrate anhand der Gewichte falsch klassifizierter Samples ermittelt. Für die verwendete Funktion $\mathbb{I}$ gilt dabei $\mathbb{I}(true) = 1$ und $\mathbb{I}(false) = 0$.

Der Algorithmus SAMME unterscheidet sich von AdaBoost durch eine einzige Gleichung. Diese stellt die Vorschrift zur Berechnung der $\alpha^{(m)}$-Koeffiziente dar, die in SAMME folgendermaßen berechnet werden:

[2] Stagewise Additive Modeling using a Multi-class Exponential loss function

Input: Eine Menge von Training-Samples mit Klassenbezeichnern $\{(x_i, c_i)|i \in \underline{n}\}$, wobei $c_i \in C = \{1, \ldots, K\}$; eine Gewichteverteilung über die Samples $W = (w_1, \ldots, w_n)$, ein schwacher Klassifikator $D^{(0)}$; Anzahl der Iterationen M

1. Initialisiere die Gewichteverteilung der Samples mit uniformer Verteilung, also $w_i = 1/n, i = 1, \ldots, n$.

2. *m=1, ..., M*

   - Trainiere den Klassifikator $D^{(m)}$ mit den Samples, unter Rücksicht auf die Gewichteverteilung W.
   - Berechne
   $$\mathrm{err}^{(m)} = \frac{\sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq D^{(m)}(x_i)\right)}{\sum_{i=1}^{n} w_i} \tag{30}$$
   - Berechne
   $$\alpha^{(m)} = \log \frac{1 - \mathrm{err}^{(m)}}{\mathrm{err}^{(m)}} + \log(K - 1) \tag{31}$$
   - Aktualisiere die Gewichte
   $$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \mathbb{I}\left(c_i \neq D^{(m)}(x_i)\right)\right), \quad i = 1, \ldots, n. \tag{32}$$
   - Normalisiere die Gewichte
   $$w_i \leftarrow w_i / \sum_{i=1}^{n} w_i, \quad i = 1, \ldots, n \tag{33}$$

3. Berechne die Endhypothese
$$C(x) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(D^{(m)}(x) = k). \tag{34}$$

Abbildung 24: SAMME Algorithmus

$$\alpha^{(m)} = \log \frac{1 - \text{err}^{(m)}}{\text{err}^{(m)}} + \log(K - 1),$$

wobei $\text{err}^{(m)}$ in Gleichung 30 der Abbildung 24 definiert ist. In der nicht-erweiterten Variante sieht die Gleichung folgendermaßen aus:

$$\alpha^{(m)} = \log \frac{1 - \text{err}^{(m)}}{\text{err}^{(m)}}.$$

Somit lässt sich für die Anzahl der Klassen $K = 2$ SAMME direkt in AdaBoost überführen. Allerdings erhält man für $K > 2$ bei SAMME für einen schwachen Klassifikator mit $err^{(m)} < \frac{K-1}{K}$ noch positive Koeffiziente $\alpha^{(m)}$, wobei bei AdaBoost die obere Schranke für $err^{(m)}$, bei der $\alpha^{(m)}$ noch positiv ist, bei 0.5 liegt. Dies hat eine Auswirkung auf die Aktualisierung der Gewichteverteilung in Gleichung 32. Ist das Sample richtig klassifiziert, so bleibt das entsprechende w_i unverändert. Die w_is der falsch klassifizierten Samples werden mit einer exponentiellen Funktion gewichtet. Ist $err^m > 0.5$, so erhält man in AdaBoost einen Koeffizient $\alpha^{(m)} < 0$. In diesem Fall werden die Gewichte (Gleichung 32) in die falsche Richtung aktualisiert, was zum Scheitern von AdaBoost führen kann (siehe Beispiel in [40]).

Vor der Verwendung in der nächsten Algorithmus-Iteration muss der neue Gewichsvektor zu einer Verteilung überführt werden. Dafür werden in Gleichung 33 die Gewichte normiert. Die Endhypothese wird durch gewichtetes Voting in Gleichung 34 ermittelt.

5.6 Adaptive Occlusion Classifier

Adaptive Occlusion Classifier ist ein Ensemble, das für die Klassifikation der Konturen mit verschiedenem Grad der Verdeckung entwickelt wurde. Es besteht aus mehreren Ebenen (Klassifikatoren), von denen jede zusätzlich geboostet werden kann. AOC ermöglicht einen lokalen Ansatz sowohl beim Training als auch im Test dadurch, dass seine Mitglieder sich auf verschiedene Verdeckungsstufen spezialisieren. Die Klassifikator-Entscheidungen werden mit dem *class-consious weighted average* Verfahren kombiniert. Hierfür werden die Gewichte mit Hilfe einer zweiten Sample-Menge und der Lösung eines Minimierungsproblems ermittelt.

Für das Training, die Ermittlung der Gewichte und die Testvorgänge werden die Konturen mit der hierarchischen Fragmentierung vorverarbeitet. Als Grundlage für das Training dient eine Sample-Menge, die ausschließlich aus den vollständigen Konturen besteht. Angenommen, dass mit der hierarchischen Fragmentierung L hierarchische Ebenen von

Fragmenten erzeugt sind. Für das Training erhalten die Konturstücke den Klassenbezeichner ihrer Stammfigur. Im nächsten Schritt werden genau L Klassifikatoren erstellt, wobei für $i \in \underline{L}$ der Klassifikator D_i mit den Samples der i-ten Ebene trainiert wird. Somit spezialisiert sich jeder Klassifikator im Ensemble auf eine Verdeckungsstufe, die mit der entsprechenden Ebene der hierarchischen Fragmentierung festgelegt wird.

Für die Ermittlung der Gewichte (Abschnitt 5.6.1) wird eine zweite Sample-Menge benötigt. Im Gegensatz zur Trainingsmenge besteht diese aus Konturen, die mit einer zufälligen Verdeckung versehen sind. Die obere Schranke für die randomisierte Verdeckung wird hier, abhängig von dem eingesetzten Verdeckungsverfahren, mit β^w_{max} bzw. γ^w_{max} bezeichnet (für die Notation siehe den Abschnitt 6.1). Im Falle der K Figurklassen und L Klassifikatoren werden mit dem Minimierungsverfahren $L \times K$ Gewichte ermittelt. Mit diesem Verfahren bezweckt man eine Menge der Gewichte zu finden, die nach der Bildung des gewichteten Durchschnittes den minimalen Fehler bei der Klassenzuordnung ergibt. Der Einsatz verdeckter Konturen für die Ermittlung der Gewichte spielt eine wichtige Rolle, da sie eine Anpassung der Funktionsweisen des Ensembles an die verdeckten Formen ermöglicht. Im nächsten Abschnitt wird das Verfahren detailliert erläutert.

5.6.1 Ermittlung der Gewichte

Wie oben beschrieben, müssen für die Verwendung des *class-consious* gewichteten Durchschnitts K Gewichte für jeden der L Klassifikatoren ermittelt werden. Dabei besteht das Ziel darin, einen solchen Vektor zu finden, dass für alle Samples der gewichtete Durchschnitt der korrekten Klasse am höchsten ausfällt. Diese Aufgabe lässt sich mit Hilfe eines Optimierungsverfahrens und einer Sample-Menge lösen, die aus teilweise verdeckten Konturen besteht.

Sei $D := (D_1, \ldots, D_L)$ das Klassifikator-Ensemble, K die Anzahl der Klassen und $X := \{(x_n, c_n) | n \in \underline{N}\}$ die Sample-Menge. Sei $l \in \underline{L}$, dann ist die Antwort-Matrix $R_l \in \mathbb{R}^{N \times K}$ des l-ten Klassifikators auf die Sample-Menge X gegeben durch:

$$R_l := \begin{pmatrix} d_{l1}(x_1) \ldots d_{lK}(x_1) \\ \ldots\ldots\ldots \\ d_{l1}(x_N) \ldots d_{lK}(x_N) \end{pmatrix}$$

Für die Ermittlung des Gewichtsvektors benötigt man eine Hilfsfunktion, die die Skalarmultiplikation und das korrekte Addieren einzelner Spalten der Matrizen $R_l,\ l \in \underline{L}$ für die

Lösung des Minimierungsproblems (siehe Gleichung 35) ermöglicht. Für ein $k \in \underline{K}$ wird sie folgendermaßen definiert:

$$f_k : \mathbb{R}^{N \times K} \to \mathbb{R}^{N \cdot K}, \quad R_l \mapsto (v_1, \ldots, v_p)^T,$$

wobei $p = N \cdot K$ und für $i \in \{1, \ldots, p\}$, $n \in \underline{N}$ gilt:

$$v_i = \begin{cases} d_{lk}(x_n), & \text{falls } \; i = N \cdot (k-1) + n \\ 0, & \text{sonst.} \end{cases}$$

Es ergibt sich z.B. für $k = 1, 2$ und K:

$$f_1(R_l) = (d_{l1}(x_1), \ldots, d_{l1}(x_N), \underbrace{0, \ldots, 0}_{N(K-1)})^T$$

$$f_2(R_l) = (\underbrace{0, \ldots, 0}_{N}, d_{l2}(x_1), \ldots, d_{l2}(x_N), \underbrace{0, \ldots, 0}_{N(K-2)})^T$$

$$f_K(R_l) = (\underbrace{0, \ldots, 0}_{N(K-1)}, d_{lK}(x_1), \ldots, d_{lK}(x_N))^T$$

Sei $r = L \cdot K$ die Anzahl der gesuchten Gewichte. Die Antworten des Klassifikator-Ensembles auf eine Sample-Menge X können in der Matrix $\hat{R} \in \mathbb{R}^{p \times r}$ zusammengefasst werden:

$$\hat{R} := (f_1(R_1), f_2(R_1), \ldots, f_j(R_i), \ldots, f_K(R_L)).$$

Für die Sample-Menge X erhält man den gesuchten Gewichtsvektor $w \in \mathbb{R}^r$ mit

$$w := (w_{11}, w_{12}, \ldots, w_{lk}, \ldots, w_{LK})^T,$$

indem man den Abstand zwischen der optimalen Antwort-Matrix R_{opt} und der entsprechend der Gleichung 28 gewichteten Antwort-Matrix $\hat{R}$ minimiert:

$$\min_{w \in \mathbb{R}^r} \|\hat{R}w - R_{opt}\|. \tag{35}$$

Der Gewichtsvektor lässt sich beispielsweise mit Hilfe des QR-Verfahrens ermitteln.

5.6.2 Zusammenfassung des Algorithmus

Im Folgenden werden die einzelnen Schritte vom Trainings- und Testvorgang des Adaptive Occlusion Classifiers zusammengefasst (siehe Abbildungen 25 und 26). Als Training-Input

(siehe Abbildung 25) sind zwei mit Klassenbezeichnern versehenen Sample-Mengen erforderlich. Die erste Sample-Menge enthält nur die vollständigen Konturen und wird für das Training der Klassifikatoren im Ensemble verwendet. Mit Hilfe der hierarchischen Fragmentierung lernt das Ensemble während des Trainings die Konturstruktur lokal. Die zweite Sample-Menge enthält Konturen zufällig verdeckter Objekte. Diese Menge dient der Ermittlung vom Gewichtsvektor, der das Ensemble an die Klassifikation teilweise verdeckter Konturen anpasst.

Um die Samples einer Testmenge T zu klassifizieren (Abbildung 26), wird wieder die hierarchische Fragmentierung angewendet. Die Klassifikator-Antworten werden entsprechend der *class-concious weighted average* für die Endentscheidung kombiniert.

Das Boosting kann im Schritt 2 vom Training eingesetzt werden. Sei M die erwünschte Anzahl der Iterationen, dann lautet der für das Boosting angepasster Schritt:

2': $l = 1, \ldots, L$
$m = 1, \ldots, M$
Trainiere den Klassifikator D_{lm} entsprechend dem Algorithmus SAMME mit den Konturfragmenten X_L, wobei jedes Fragment der Menge X_L mit dem Klassenbezeichner der zugehörigen Gesamtkontur trainiert wird.

Nach dem Training stehen auf jeder Ebene $l \in \underline{L_\sqcup}$ jeweils M Klassifikatoren zur Verfügung. Nach Gleichung 34 kann die Entscheidung für die jeweilige Ebene berechnet werden.

Adaptive Occlusion Classifier (Training)

Input: Zwei Sample-Mengen $X = \{(x_n, c_n) | n \in \underline{N}\}$ und $Y = \{(y_m, c_m) | m \in \underline{M}\}$, $N, M \in \mathbb{N}$.

Training:

1. Wende das hierarchische Fragmentierungsverfahren auf die Menge der Training-Samples X an. Sei L die Anzahl der dabei entstandenen Ebenen und seien $(X_1, \ldots, X_L)$ die ensprechenden Fragmente-Mengen.

2. $l = 1, \ldots, L$
   Trainiere den Klassifikator D_l mit den Merkmalsvektoren der Konturfragmente X_l, wobei jedes Fragment der Menge X_l mit dem Klassenbezeichner der entsprechenden Gesamtkontur versehen wird.

3. Wende das hierarchische Fragmentierungsverfahren auf die Sample-Menge Y an und erhalte die Fragmente-Mengen $(Y_1, \ldots, Y_L)$.

4. $l = 1, \ldots, L$
   Berechne den Antwort-Vektor $D_l(y)$ für jedes Sample $y \in Y$. Hierfür addiere die Antwort-Vektoren für alle aus y erstellten Fragmente auf Ebene l. Aufbauend darauf, berechne die Antwort-Matrix R_l des Klassifikators D_l.

5. Gegeben sind die Matrizen $(R_1, \ldots, R_L)$, berechne die Matrix $\hat{R}$.

6. Berechne den Gewichtsvektor w als Lösung des linearen Optimierungsproblems

$$\min_{w \in \mathbb{R}^r} \|\hat{R}w - R_{opt}\|.$$

Abbildung 25: Adaptive Occlusion Classifier Ensemble Training und Ermittlung des Gewichtsvektors.

Adaptive Occlusion Classifier (Test)

Input: Eine Menge der Test-Samples T, Gewichtsvektor w.

Test:

1. Berechne die Fragment-Mengen $(T_1, \ldots, T_L)$ der Testmenge T mit dem hierarchischen Fragmentierungsverfahren.

2. $l = 1, \ldots, L$
   Berechne den Antwort-Vektor $D_l(x)$ für jedes Test-Sample $x \in T$. Hierfür addiere die Antwort-Vektoren für alle aus x erstellten Fragmente auf Ebene l.

3. Verwende den Gewichtsvektor w

   $$w := (w_{11}, w_{12}, \ldots, w_{lk}, \ldots, w_{LK}),$$

   um für ein Test-Sample $x \in T$ die Antworten für die Klasse $j \in \underline{K}$ nach der Methode *class-conscious weighted average* zu gewichten:

   $$\mu_j(x) = \sum_{i=1}^{L} w_{ij} d_{ij}(x).$$

4. Der Klassenbezeichner wird entsprechend dem Maximum der $\mu_j(x)$s zugeordnet:

   $$\arg\max_{j \in K} \mu_j(x).$$

Abbildung 26: Adaptive Occlusion Classifier Ensemble Test.

6 Experimentelle Ergebnisse

Besonders wichtig für die Funktionsweise des Adaptive Occlusion Classifiers sind die folgenden Bereiche:

- Wahl des Merkmalraums
- Dimension des Merkmalraums
- Anzahl der Training-Samples
- Klassifikationsergebnisse bei unterschiedlichem Verdeckungsgrad
- Beitrag von AdaBoost
- Training mit künstlich generierten Objektbildern

In diesem Kapitel werden als Erstes Testergebnisse für unterschiedliche Dimension von Merkmalsvektoren aus FD, ZM und PZM vorgestellt. Die nächste Versuchsreihe zeigt die Abhängigkeit der Klassifikationsfehlerrate von der Sample-Anzahl beim Training und der Gewichtsvektor-Ermittlung. Anschließend wird die Robustheit des Klassifikator-Ensembles gegen die Verdeckungsgrad-Veränderung bei der Klassifikation getestet. Einige Ergebnisse werden zur Abhängigkeit der Klassifikationsgüte von der Anzahl der AdaBoost-Iterationen präsentiert. Zum Abschluss werden Experimente zum Klassifikator-Training und -Testen mit künstlichen Bildern durchgeführt. Innerhalb der Versuchsreihe werden zwei Mengen künstlich generierter Objektbilder verwendet, die sich durch den Grad perspektivischer Verzerrung unterscheiden. Hierbei wird u.a. nach einer Möglichkeit gesucht den Trainingsvorgang zu automatisieren. Das Klassifikator-Training mit künstlichen Bildern erfordert anstatt eines manuell aufgenommenen Datenpools ausschließlich eine Angabe der Einstellungen und Parameter zur Generierung einer Sequenz künstlicher Bilder. Innerhalb dieser Arbeit benutzt man zum Erstellen künstlicher Bilder den *Persistence of Vision Raytracer (POV-Ray)* [3, 2]. Mit Hilfe dieser Applikation lassen sich realitätsnahe dreidimensionale Szenen mit diversen Lichtquellen, anpassbaren Lichtverhältnissen und Kameraeinstellungen leicht erstellen.

Für die Versuche wird ein Datenpool von 1000 Kamerabildern und 1600 mit POV-Ray generierten Bildern verwendet. Für jede der fünf Figuren Brücke, Zylinder, Halbzylinder, Quader und Prisma ist eine gleiche Anzahl von Bildern vorhanden. Der Datenpool besteht

aus Bildern der Figuren, deren Konturen durch die Blickwinkelveränderung der Kamera bei der Aufnahme bzw. durch POV-Ray-Einstellungen perspektivisch verzerrt sind. Bei allen Versuchen innerhalb dieser Arbeit verwendet man paarweise verschiedene Test-, Trainings- und Gewichtsermittlungsmengen, wobei der Datenpool bei jedem Versuch zufällig permutiert wird. Alle im Test und Training eingesetzten Sample-Mengen enthalten die gleiche Anzahl der Konturen jeder Klasse. Die Verdeckungen werden synthetisch aus vollständigen Konturen und Schwarzweißbildern generiert.

6.1 Notation für die Verdeckung im Trainings- und Testvorgang

Die Notation für die Verdeckung, die im Abschnitt 5.2.1 eingeführt ist, wird hier weiter verwendet. Somit wird die randomisierte Verdeckung der Konturen in einer Sample-Menge vereinfachend durch die Angabe der oberen Schranke β_{max} in der ersten Methode vorgegeben und γ_{max} in der zweiten. Eine konstante Verdeckung wird für eine Sample-Menge durch die Angabe der Konstante β_{const} bzw. γ_{const} vorgegeben. Die Werte aller oben beschriebenen Parameter liegen im Intervall $[0, 1]$. Im Kontext vom Ensemble-Test und -Training differenziert man in den weiteren Abschnitten zusätzlich zwischen den Verdeckungsparametern für die Menge zur Ermittlung des Gewichtsvektors und für die Testmenge. Verwendet man die obere Schranke für die Menge zur Ermittlung der Gewichte, so wird sie mit β^w_{max} bzw. γ^w_{max} bezeichnet. Gibt die obere Schranke eine randomisierte Verdeckung der Testmenge an, so verwendet man β^t_{max} bzw. γ^t_{max}. Analog wird die Notation für die Angaben konstanter Verdeckung benutzt.

6.2 Experimente zum Merkmalraum und seiner Dimension

In diesem Abschnitt werden Klassifikationsergebnisse vorgestellt, bei denen Fourier-Deskriptoren, Zernike- und pseudo Zernike-Momente zur Merkmalsextraktion verwendet werden. Das Ziel der Experimente besteht darin, die optimale Dimension des Merkmalraumes zu finden und die drei Merkmalsextraktionsverfahren miteinander zu vergleichen. Für alle Tests innerhalb dieser Versuchsreihe werden ausschließlich echte Kamerabilder eingesetzt.

6.2.1 Experimentelle Rahmenbedingenung

In diesem Versuch wird die Abhängigkeit der Fehlerrate von der Dimension des Merkmalsvektors ermittelt, deshalb müssen die restlichen Parameter des AOC-Ensembles festgehalten werden. Hierbei werden für das Ensemble-Training 370 vollständige Konturen aus dem Datenpool genommen. Für die Ermittlung des Gewichtsvektors verwendet man 250 Konturen mit $\beta^w_{max} = 0.5$ bzw. $\gamma^w_{max} = 0.3$. Der dem β^w_{max} entsprechenden Wert von γ^w_{max} wird nach der Gleichung 23 aus dem Abschnitt 5.2.1 ermittelt. Getestet werden 250 Konturen, 50 für die jeweilige Figurklasse mit $\beta^t_{max} = 0.5$ und $\gamma^t_{max} = 0.3$.

Für die hierarchische Fragmentierung der Konturen wird die maximale Anzahl der Stützstellen auf 35 gesetzt. Experimentell erwies sich diese für die vorhandenen Konturdaten als optimal. Bei der Gauß-Glättung verwendet man das empirisch ermittelte Optimum von $\sigma = 5$. Vor der Bearbeitung durch den Merkmalsextraktor werden alle Konturdaten auf eine Anzahl der Pixel gleich 64 normiert. Diese hat sich wiederum experimentell unter anderen Zweierpotenzen auch hinsichtlich des Rechenzeitbedarfs als die geeignetste gezeigt. Die in diesem Paragraphen angegebenen Werte für die hierarchische Fragmentierung, Gauß-Verfahren usw. werden in allen weiteren Versuchen beibehalten.

6.2.2 Anzahl der Fourier-Deskriptoren

Nach der Konturvorverarbeitung, FFT und dem Normalisierungsvorgang erhält man von dem Merkmalsextraktor für jedes Fragment den Vektor $f := (f_1, \ldots, f_{63})$ der normalisierten Fourier-Deskriptoren. Der erste Test wurde mit einer aufsteigenden Anzahl der Fourier-Deskriptoren durchgeführt. Für diese Versuchsreihe werden die Vektoren

$$(f_1, \ldots, f_i), \quad i = 2, \ldots, 63 \tag{36}$$

verwendet. Die Absicht dabei ist, die am besten geeignete Anzahl von Deskriptoren zu finden. Experimenell wurde ermittelt, dass für die vorhandene Komplexität der Konturen die Deskriptoren ab der 15. Ordnung in den meisten Fällen nicht mehr zur Klassifikation beitragen. In Abbildung 27 beträgt i maximal 18. Der obere Graph dieser Abbildung stellt die Testergebnisse mit der Methode Kontur-Verdeckung dar, der untere Graph mit der Methode Flächen-Verdeckung. An der X-Achse der beiden Graphen ist die Vektordimension i eingetragen. An der Y-Achse kann man die über zehn Durchläufe gemittelten Klassifikationsergebnisse und die mit den Fehlerbalken dargestellte Standardabweichung sehen.

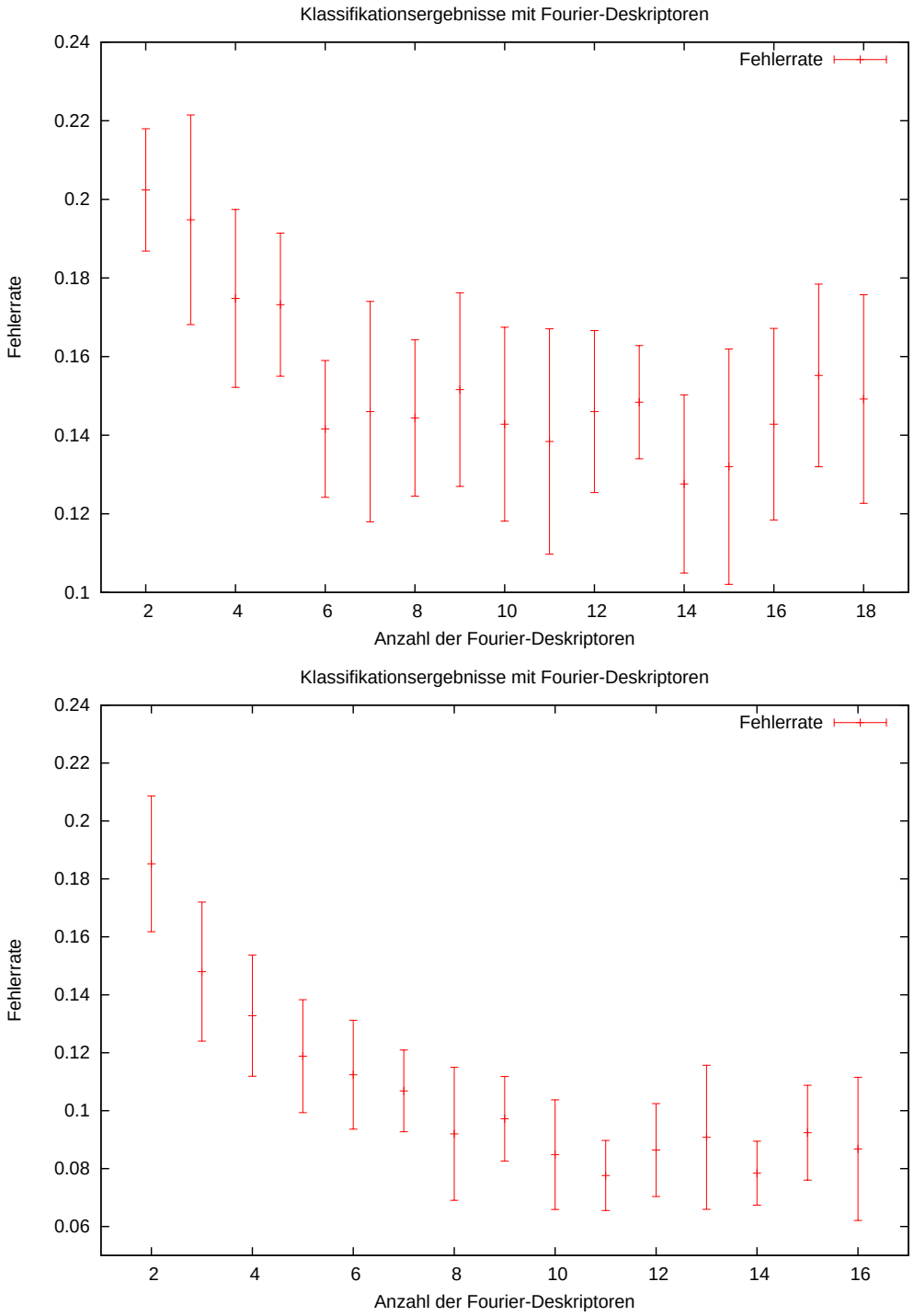


Abbildung 27: Abhängigkeit der Fehlerrate von der FD-Anzahl; Standardabweichung (1σ) ist mit den Fehlerbalken dargestellt; randomisierte Kontur-Verdeckung $\beta^t_{max} = 0.5$ (oben), randomisierte Flächen-Verdeckung $\gamma^t_{max} = 0.3$ (unten).

Die Standardabweichung wird hier und in den weiteren Experimenten nach der folgenden Vorschrift berechnet:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2},$$

wobei $\bar{x}$ den Mittelwert der Stichproben x_i bezeichnet:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

In Abbildung 27 kann man erkennen, dass der Klassifikationsfehler für 2 FD mit beiden Verfahren bei ungefähr 20 Prozent liegt und mit einer Erhöhung der Dimension bis auf 6 (oben) und 8 (unten) monoton sinkt. Eine weitere Steigerung zeigt keine deutliche Auswirkung auf die Fehlerrate, die zwischen 13 und 16 im ersten Graph und zwischen 8 und 10 Prozent im zweiten Graph schwankt. Innerhalb dieser Versuchreihe liegt das Minimum des gemittelten Fehlers bei ca. 13 bzw. 8 Prozent.

6.2.3 Anzahl der Zernike- und pseudo Zernike-Momente

Als Nächstes werden die Zernike- und pseudo Zernike-Momente für die Merkmalsextraktion eingesetzt. Der Vektor der untersuchten Momente setzt sich wie folgt zusammen:

$$m := (m_{20}, m_{22}, m_{31}, m_{33}, m_{40}, m_{42}, m_{51}, m_{53}, m_{60}, m_{62}, m_{71}, m_{73}, m_{80}, m_{82}, m_{91}, m_{93}). \tag{37}$$

Der Vektor m enthält jeweils zwei Momente der Ordnung 2 bis 9, wobei hier für die Berechnung von Momenten gleicher Ordnung das Verfahren aus dem Abschnitt 2.4.2 eingesetzt werden kann. Für eine Kontur c der Länge L, eine Menge $D = \underline{L-1}_{\mid}$ und eine Menge $P = c(D)$ wird ein m_{nm} folgendermaßen berechnet:

$$m_{nm} = |Z_{nm}| = \left| \frac{n+1}{\pi} \sum_{(x,y)\in P} [V_{nm}(x,y)]^* \right|.$$

Dabei gilt für Zernike-Momente:

$$V_{nm}(x,y) = V_{nm}(\rho,\theta) = R_{nm}(\rho)\exp(im\theta)$$

und für pseudo Zernike-Momente entsprechend:

$$V_{nm}(x,y) = V'_{nm}(\rho,\theta) = R'_{nm}(\rho)\exp(im\theta).$$

Vor der Berechnung wird die Kontur c entsprechend des Abschnitts 2.5 normiert und auf den Einheitskreis abgebildet. Die Anzahl der Punkte in einer Kontur vor der Berechnung der PZM bzw. ZM beträgt wie bei der Fourier-Transformation 64. Dies ist u.a. behilflich beim Vergleich der Klassifikationsergebnisse von drei Merkmalsextraktionsverfahren. Tests mit unterschiedlichen Reihenfolgen der Momente in m haben vergleichbare Fehlerwerte ergeben. Aus diesem Grund wird hier bei allen Versuchen mit einer steigenden Anzahl der Momente die Reihenfolge aus Gleichung 37 beibehalten, d.h., dass die Merkmalsvektoren aus ZM bzw. PZM für die Dimensionen $i = 2, \ldots, 16$ analog zu den FD in Gleichung 36 gebildet werden.

In Abbildung 28 werden die Ergebnisse der Versuchsreihe mit einer wachsenden Dimension des ZM-Vektors dargestellt. Hier ist der Klassifikationsfehler und die Standardabweichung für $\beta^t_{max} = 0.5$ (oben) und $\gamma^t_{max} = 0.3$ (unten) zu sehen. An der X-Achse ist analog zur letzten Versuchsreihe die Dimension i des Merkmalsvektors abgebildet. An der Y-Achse ist die über fünf Durchläufe gemittelte Fehlerrate eingetragen. Für $i = 2$ beträgt der Fehler in beiden Graphen ca. 35 Prozent. Ähnlich wie bei FD kann man bei einer Erhöhung der Dimension bis auf 9 für das erste Verdeckungsverfahren und bis auf 7 für das zweite Verdeckungsverfahren eine monotone Verbesserung der Klassifikationsgenauigkeit beobachten. Bei einer Steigerung ab der 9. bzw. 12. Dimension kann man kein deutliches Sinken des Durchschnittsfehlers mehr erkennen. Er schwankt zwischen 14 und 16 bzw. zwischen 7 und 8 Prozent. Das Minimum liegt bei 13 Prozent für die Kontur-Verdeckung und bei 7 Prozent für die Flächen-Verdeckung.

Die pseudo Zernike-Momente in Abbildung 29 zeigen für die beide Verdeckungsverfahren eine ähnliche Dynamik wie die Zernike-Momente und die Fourier-Deskriptoren. Für $i = 2, \ldots, 7$ wird ein monotones Sinken der Fehler um ca. 30 Prozent beobachtet. Für die Kontur-Verdeckung liegt das Minimum bei ungefähr 13 Prozent, für die Flächen-Verdeckung bei ca. 8 Prozent.

6.2.4 Vergleich der Merkmalsextraktionsverfahren

Vergleicht man die durchschnittlichen Fehlerwerte der Fourier-Deskriptoren, der Zernike-, bzw. pseudo Zernike-Momente (siehe Abbildung 30), so erkennt man, dass bei den durchgeführten Experimenten mit Kontur-Verdeckung (oben) ab der 9. Dimension die FD und ZM vergleichbar gute Ergebnisse liefern. Hierbei liegen die Fehlerwerte für PZM meistens etwas höher. Mit Flächenverdeckung bekommt man durch die Verwendung der ZM die

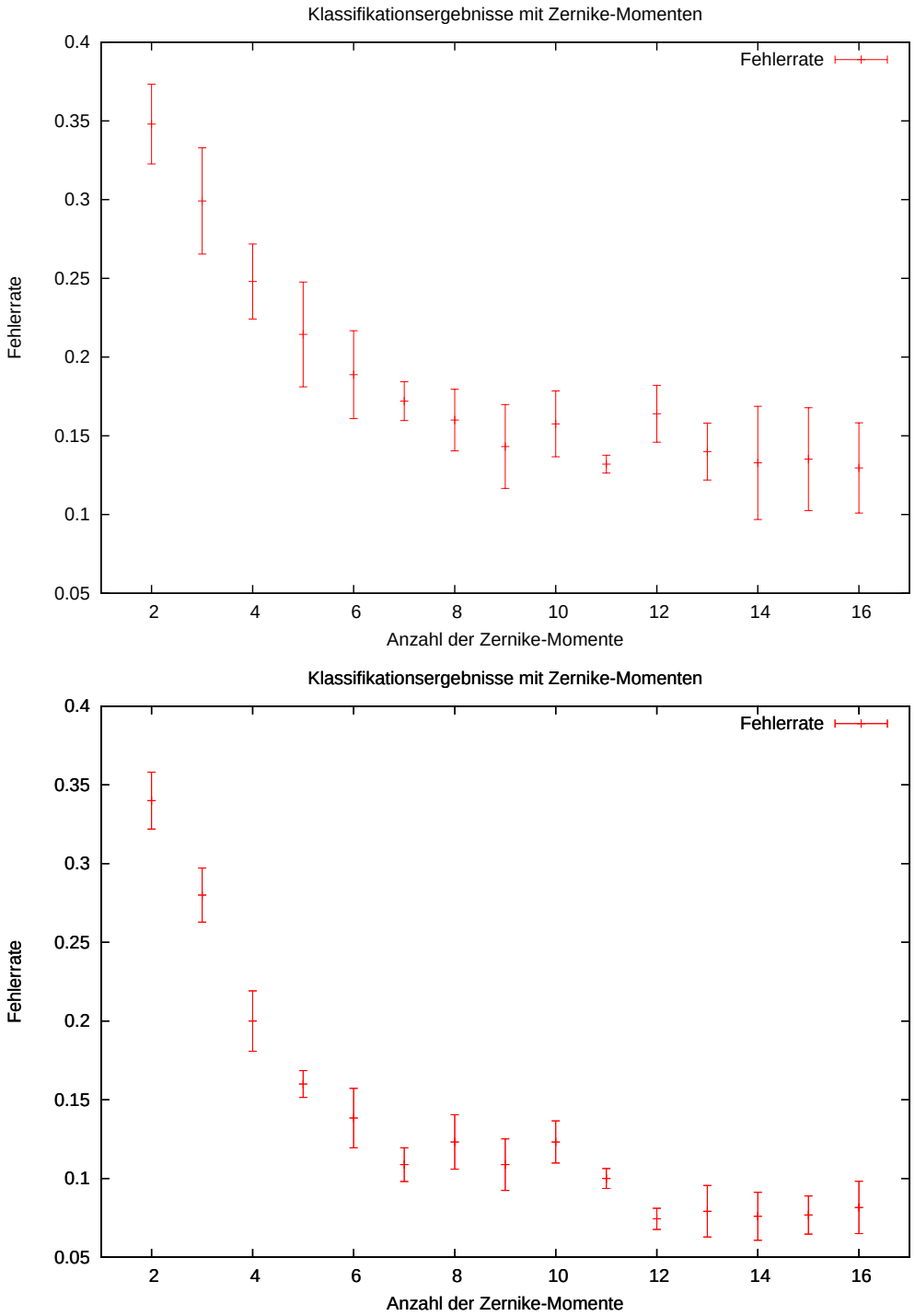


Abbildung 28: Abhängigkeit der Fehlerrate von der ZM-Anzahl; Standardabweichung (1σ) ist mit Fehlerbalken dargestellt; Testergebnisse für $\beta^t_{max} = 0.5$ (oben) und für $\gamma^t_{max} = 0.3$ (unten).

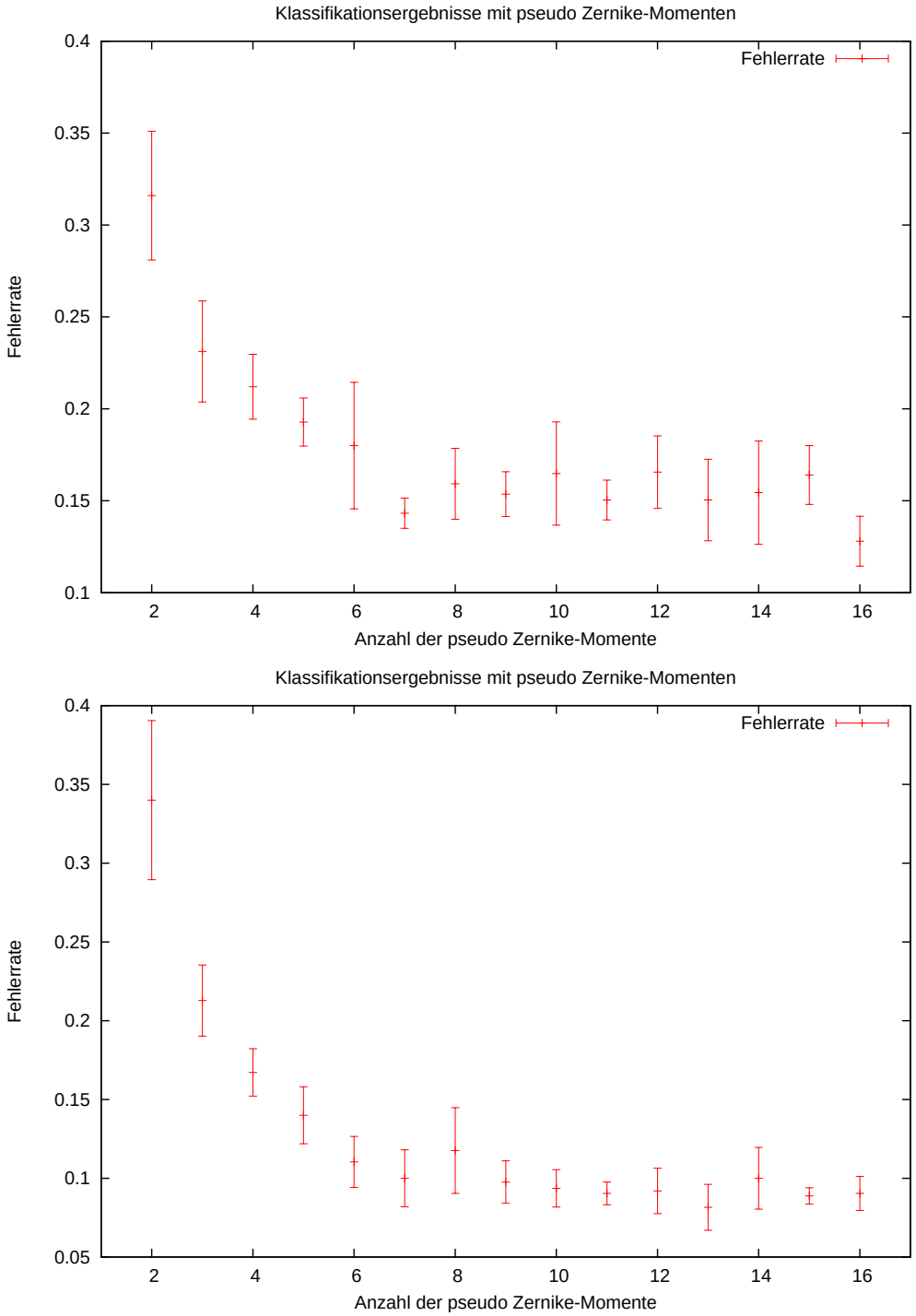


Abbildung 29: Abhängigkeit der Fehlerrate von der PZM-Anzahl; Standardabweichung (1σ) ist mit Fehlerbalken dargestellt; Testergebnisse für $\beta^t_{max} = 0.5$ (oben) und $\gamma^t_{max} = 0.3$ (unten).

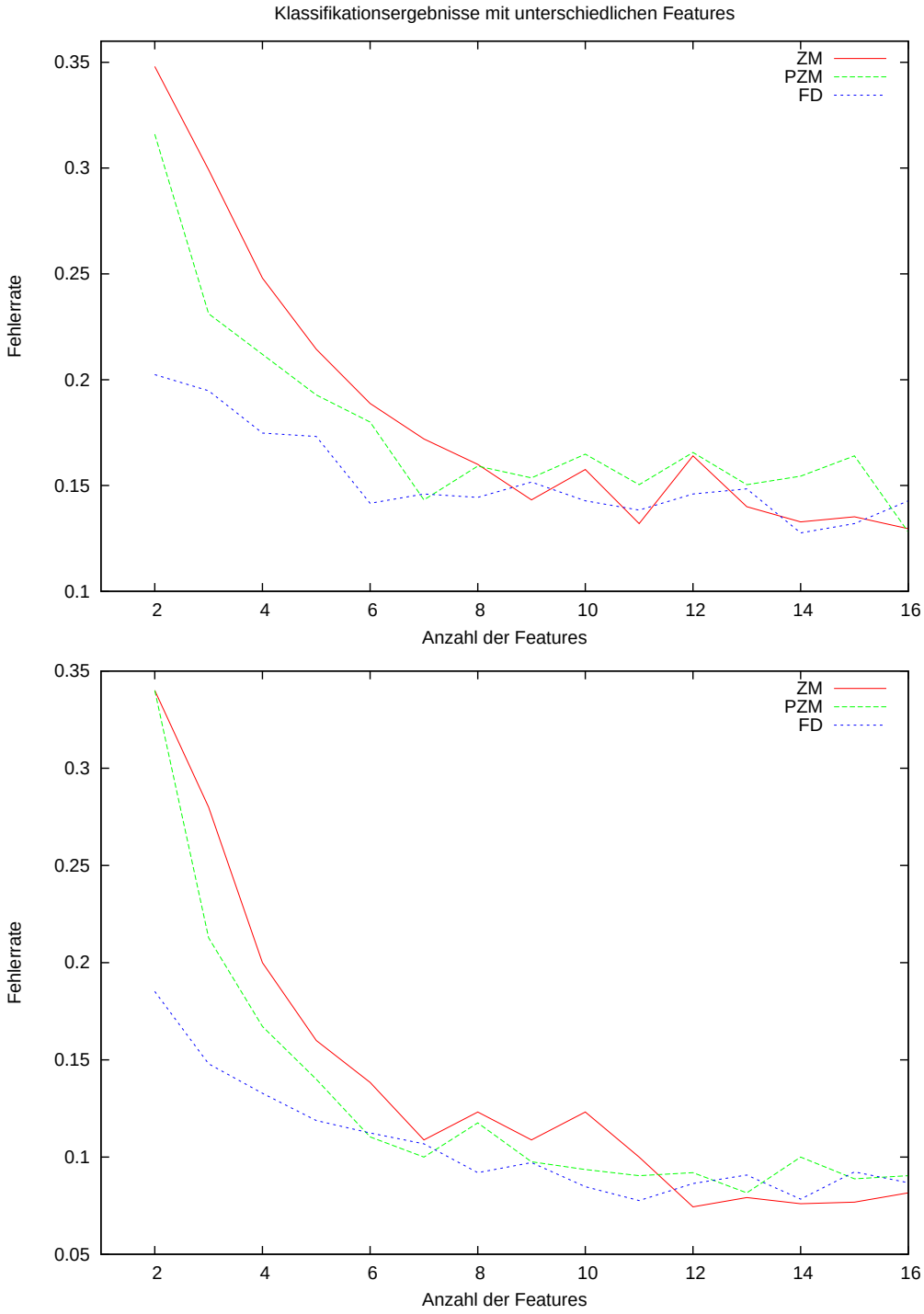


Abbildung 30: Durchschnittliche Fehlerrate für Fourier-Deskriptoren (FD), Zernike-Momente (ZM) und pseudo Zernike-Momente (PZM); Zusammenfassung der Ergebnisse aus Abbildungen 27, 28 und 29.

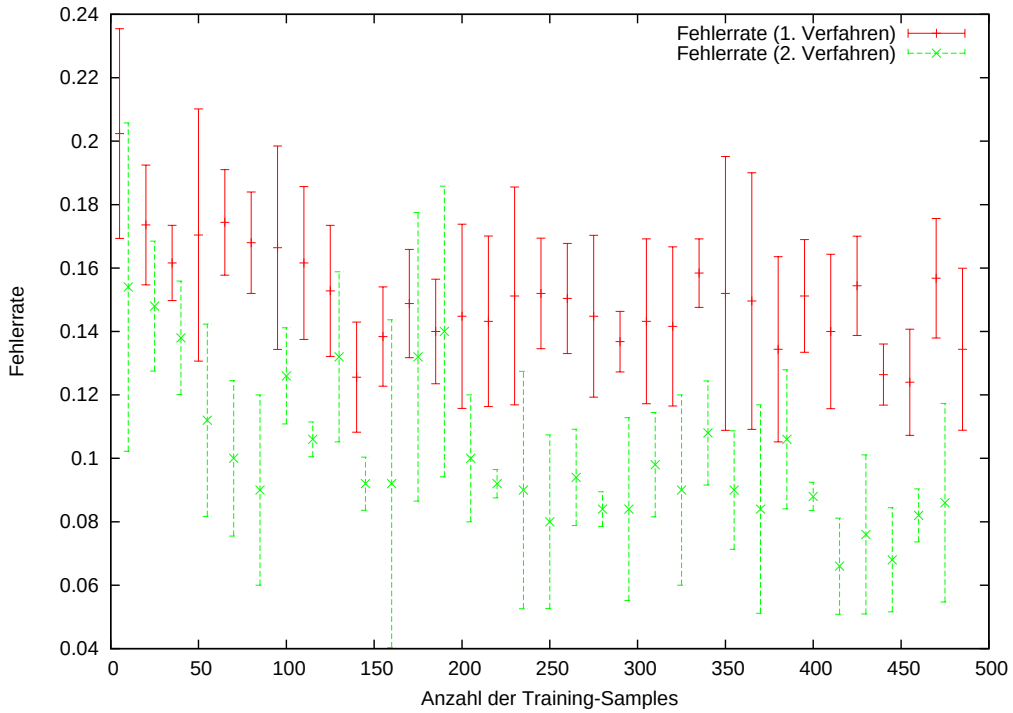


Abbildung 31: Fehlerrate in Abhängigkeit zur steigenden Anzahl der Training-Samples; Testergebnisse für $\beta^t_{max} = 0.5$ (rot) und für $\gamma^t_{max} = 0.3$ (grün); Standardabweichung (1σ) ist mit Fehlerbalken dargestellt.

niedrigsten Fehlerraten ab der 12. Dimension. Die Fourier-Deskriptoren und die pseudo Zernike-Momente schneiden in den Tests vergleichbar gut ab.

6.3 Experimente zum Klassifikator-Training

Wie im Kapitel 5 beschrieben, benötigt das AOC-Ensemble zum Training zwei Sample-Mengen. Mit der ersten Konturmenge werden die einzelnen Klassifikatoren gelernt, mit der zweiten Menge randomisiert verdeckter Konturen wird der Gewichtsvektor ermittelt. In den nächsten Abschnitten werden die Ergebnisse der Versuche mit einer wachsenden Anzahl der Samples in den beiden Mengen präsentiert. Ein FD-Vektor mit $i = 14$ wird hier im Trainings- und im Testvorgang verwendet.

6.3.1 Anzahl der Training-Samples

Im ersten Versuch wird getestet, wie die Anzahl der Samples im Ensemble-Training die Klassifikationsergebnisse beeinflusst. In Abbildung 31 sieht man die Abhängigkeit der Klas-

sifikationsfehlerrate von der Anzahl der Training-Samples. Hier wird die Anzahl der verwedeten Konturen (X-Achse) von 5 bis auf 500 erhöht, wobei alle anderen Parameter konstant gehalten werden. Die über fünf Durchläufe gemittelte Fehlerrate bei der Klassifikation von 250 Samples mit $\beta^t_{max} = 0.5$ (oben) und $\gamma^t_{max} = 0.3$ (unten) ist auf der Y-Achse eingetragen. In dieser Versuchsreihe wird die Menge zur Ermittlung des Gewichtsvektors mit $\beta^w_{max} = 0.5$ bzw. $\gamma^w_{max} = 0.3$ konstant bei 250 Samples gehalten. Mit unterschiedlichen Farben werden in Abbildung 31 die Ergebnisse für die beiden Verdeckungsverfahren eingezeichnet: Rot für die Kontur-Verdeckung und Grün für die Flächen-Verdeckung. Für die beiden Verdeckungsverfahren ist nur anfänglich eine schwache Verbesserung der Ergebnisse zu erkennen. Für das erste Verfahren schwankt der durchschnittliche Fehler ab 150 Training-Samples zwischen 12 und 16 Prozent. Für das zweite bleibt ab 250 Samples der Fehler im Bereich von 7 bis 11 Prozent.

6.3.2 Anzahl der Samples zur Ermittlung des Gewichtsvektors

Für die Ermittlung des Gewichtsvektor (Abschnitt 5.6.1) wird zusätzlich zu den Training-Samples eine Menge randomisiert verdeckter Konturen benötigt. Dafür verwendet man in diesem Test 5 bis 400 Konturen mit $\beta^w_{max} = 0.5$ bwz. $\gamma^w_{max} = 0.3$. Die Trainingsmenge enthält konstant 350 vollständige Konturen, die Testmenge 250 Konturen mit $\beta^t_{max} = 0.5$ bzw. $\gamma^t_{max} = 0.3$. In Abbildung 32 sieht man die resultierenden Fehlerraten für eine wachsende Anzahl der Samples zur Gewichtsvektor-Ermittlung. Für jeden Punkt wird der Mittelwert über fünf Durchläufe berechnet. Die Standardabweichung ist mit Fehlerbalken dargestellt. Mit der Farbe Rot sind wiederum die Klassifikationsergebnisse von Kontur-Verdeckung gezeichnet, mit Grün von Flächenverdeckung. In dieser Abbildung kann man wiederum erkennen, dass nur anfänglich eine Verbesserung der Klassifikationsgenauigkeit stattfindet. Bei einer weiteren Steigerung der Sample-Anzahl verändert sich die Fehlerrate nicht mehr erheblich.

6.3.3 Zusammenfassung für das Training und die Gewichtsermittlung

In diesem Abschnitt werden Testergebnisse vorgestellt, die eine Übersicht über die Abhängigkeit der Fehlerrate von der Sample-Anzahl in den beiden oben beschriebenen Mengen bieten. In Abbildung 33 ist an der X-Achse der beiden Graphen die Anzahl der Samples in der Trainingsmenge eingetragen. Die Y-Achse bezeichnet die Anzahl der Samples, die für

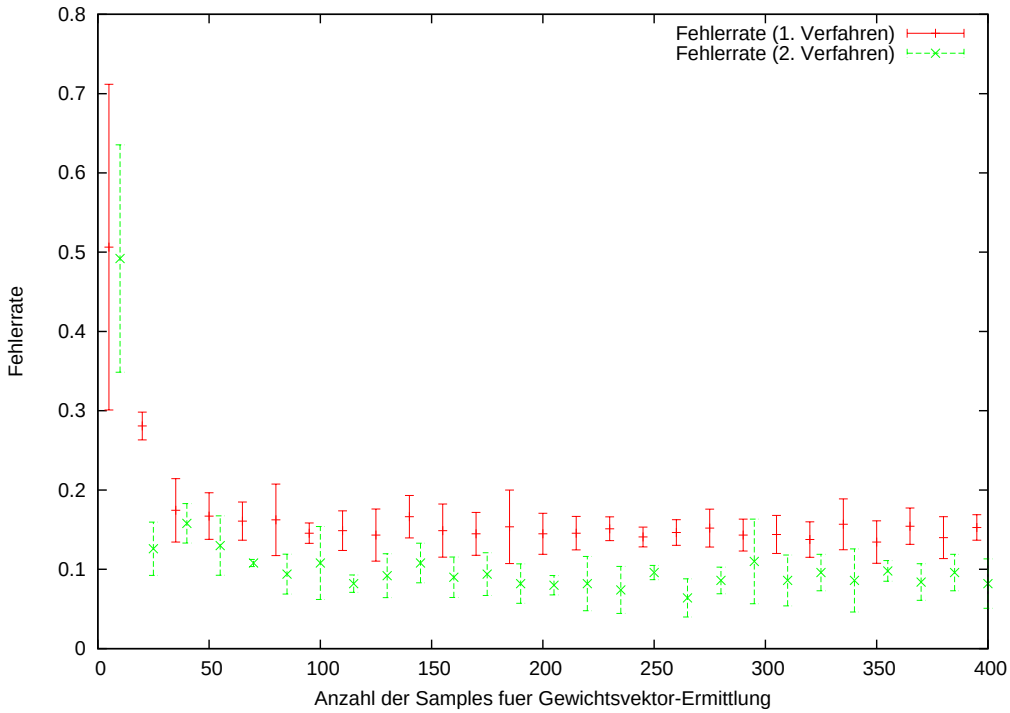


Abbildung 32: Fehlerrate in Abhängigkeit von der Sample-Anzahl in der Menge zur Ermittlung des Gewichtsvektors. Standardabweichung (1σ) ist dargestellt mit Fehlerbalken; Kontur-Verdeckung β^t_{max} (rot) und Flächen-Verdeckung mit γ^t_{max} (grün).

die Ermittlung des Gewichtsvektors verwendet wird. Die Z-Achse stellt die Fehlerrate der Klassifikation dar. Oben sind die Ergebnisse für die Kontur-Verdeckung abgebildet, unten für die Flächen-Verdeckung. In diesem Versuch wird die Sample-Anzahl in beiden Mengen von 25 auf 300 Samples erhöht; β^w_{max} beträgt 0.5 und der entsprechender γ^w_{max} beträgt 0.3. Die Testmenge besteht aus 100 Samples mit $\beta^t_{max} = 0.5$ bwz. $\gamma^t_{max} = 0.3$. In Abbildung 33 kann man die gleiche Dynamik wiedererkennen, die in den Abbildungen 31 und 32 zu sehen ist. Kaum eine Auswirkung auf die Fehlerrate zeigt eine Erhöhung der Anzahl von Training-Samples von 25 auf 300. Eine starke Verbesserung der Ergebnisse ist anfänglich mit einer Steigerung der Sample-Anzahl zur Gewichtsvektorermittlung verbunden. Eine weitere Erhöhung bewirkt keine erhebliche Verbesserung der Klassifikationsergebnisse. Hierbei schwankt der Fehler zwischen 10 und 20 Prozen für das erste Verfahren und zwischen 5 und 15 Prozent für das zweite Verfahren. Dies kann man anhand einiger Beispiele in Abbildung 34 erkennen. Sie stellt eine zweidimensionale Projektion der Daten aus Abbildung 33 für 150, 200 und 250 Training-Samples dar. Ein Teil der Klassifikationsfehler

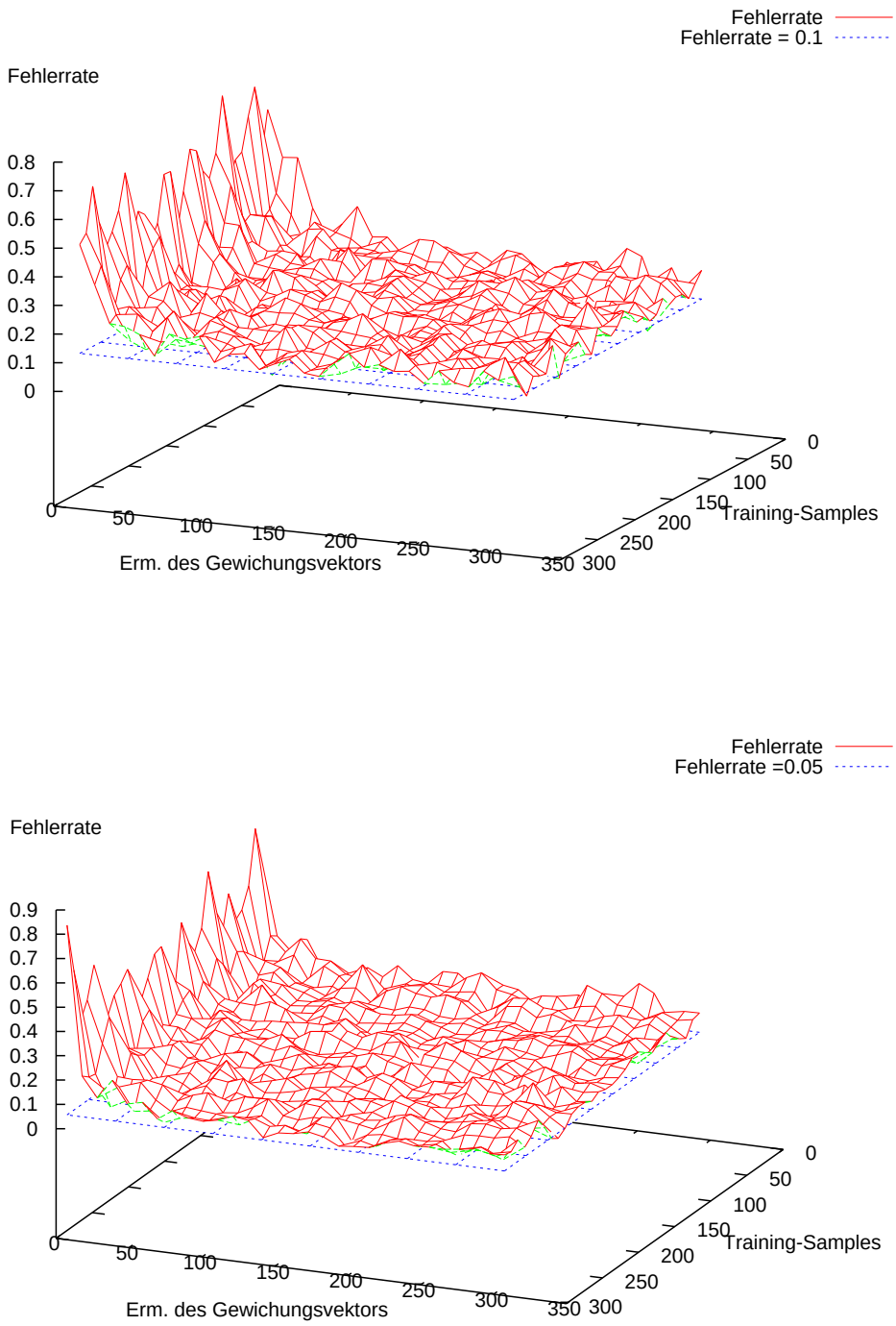


Abbildung 33: Fehlerrate in Abhängigkeit von der Sample-Anzahl in der Trainings- und der Gewichtsermittlungsmenge. Testergebnisse für $\beta^t_{max} = 0.5$ (oben) und für $\gamma^t_{max} = 0.3$ (unten).

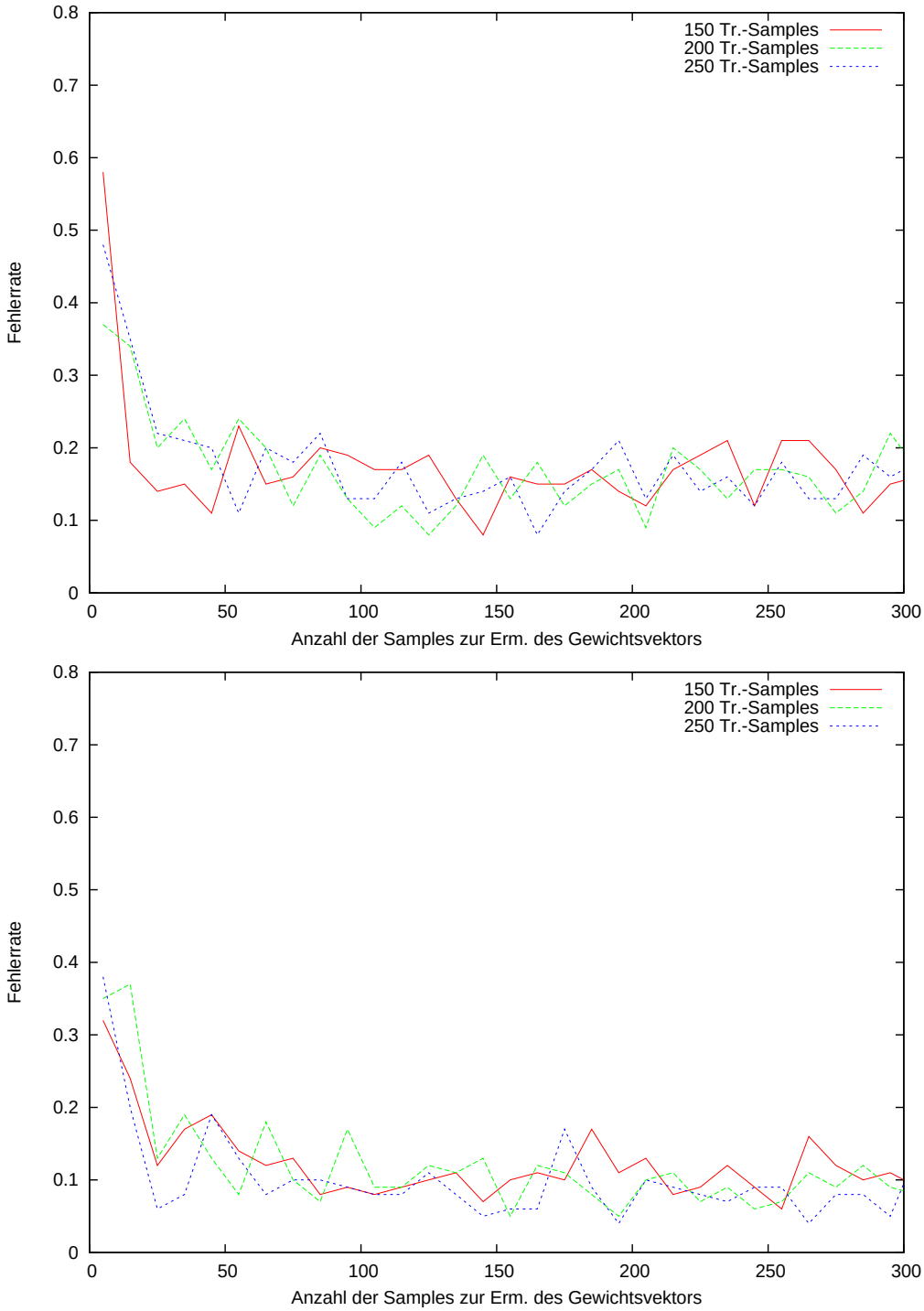
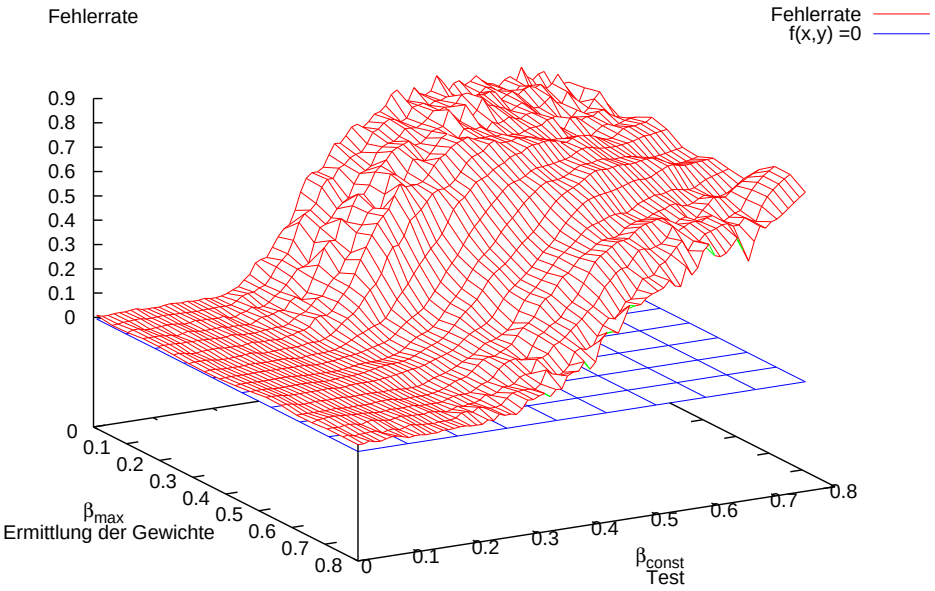


Abbildung 34: Klassifikationsfehlerrate in Abhängigkeit von der Sample-Anzahl zur Gewichtsvektorermittlung; die Anzahl von Training-Samples beträgt 150 (rot), 200 (grün), 250 (blau); Kontur-Verdeckung (oben), Flächen-Verdeckung (unten);

wird dadurch verursacht, dass eine eindeutige Zuordnung teilweise verdeckter Konturen, auch mit dem menschlichen Auge, nicht immer möglich ist.

6.4 Klassifikationsrobustheit bei variierendem Verdeckungsgrad

In den vorherigen Versuchen wurde der Verdeckungsgrad der Test- und der Gewichtsermittlungsmenge immer auf 0.5 und 0.3 festgelegt. Dieser Abschnitt bietet einen Überblick über die Klassifikationsergebnisse bei einem variablen Verdeckungsgrad. Es wird sowohl an dem Verdeckungsgrad der Testkonturen als auch an dem Verdeckungsgrad der Samples zur Ermittlung des Gewichtsvektors variiert. Sei $V = [0, 0.8]$, dann umfasst die Versuchsreihe die Tests für $(\beta^w_{max}, \beta^t_{const}) \in V \times V$ bzw. $(\gamma^w_{max}, \gamma^t_{const}) \in V \times V$, wobei der erste Parameter die obere Schranke der Verdeckung bei der Gewichtsvektorermittlung und der zweite die konstante Verdeckung der Test-Samples angibt. Die Abbildung 35 stellt somit die Funktion dar, die die beiden Verdeckungsparamter auf die Klassifikationsfehlerrate abbildet. In dieser Versuchsreihe verwendet man 350 Sample für das Ensemble-Training, 250 zur Ermittlung des Gewichtsvektors und 250 Samples für den Test. In allen drei Mengen ist die Anzahl der Samples für die jeweilige Figur-Klasse gleich. In Abbildung 35 werden die Testergebnisse für Kontur-Verdeckung (oben) und Flächen-Verdeckung (unten) präsentiert. Auf der X-Achse der Graphen ist β^w_{max} bzw. γ^w_{max} eingetragen. Die Y-Achse bezeichnet die konstante Verdeckung der Testdaten β^t_{const} bwz. γ^t_{const}. In diesem Abschnitt wird innerhalb der Testmenge keine randomisierte Verdeckung verwendet. Die Z-Achse bezeichnet die Fehlerrate. In Abbildung 35 kann man beobachten, dass sich mit der Erhöhung von β^w_{max} (oben) bzw. γ^w_{max} (unten) die Klassifikationsgenauigkeit insgesamt verbessert. Die Abbildung 36 stellt eine zweidimensionale Projektion der Daten aus Abbildung 35 für drei ausgewählten Werte von β^w_{max} bzw. γ^w_{max} dar. An der X-Achse ist die Verdeckungsrate β^t_{const} bzw. γ^t_{const} eingetragen, die Y-Achse gibt die Klassifikationsfehlerrate an. Die unterschiedlichen Farben bezeichnen $\beta^w_{max} = 0.01, 0.4, 0.8$ (oben) bzw. $\gamma^w_{max} = 0.01, 0.4, 0.8$ (unten). In Abbildungen 35 und 36 kann man erkennen, dass eine Erhöhung von β^w_{max} bzw. γ^w_{max} im Intervall $[0, 0.8]$ die Klassifikationsfehlerrate bei größeren Verdeckungsgraden der Test-Samples stark reduziert. Gleichzeitig hebt sich die Fehlerrate bei fast vollständigen Konturen leicht an. Das lässt sich damit erklären, dass bei der Optimierung der Gewichte auf die Klassifikation bis zu 80 Prozent verdeckter Konturen die auf vollständigen und fast vollständigen Konturen spezialisierenden Ensemble-Mitglieder weniger Gewichtung bei der



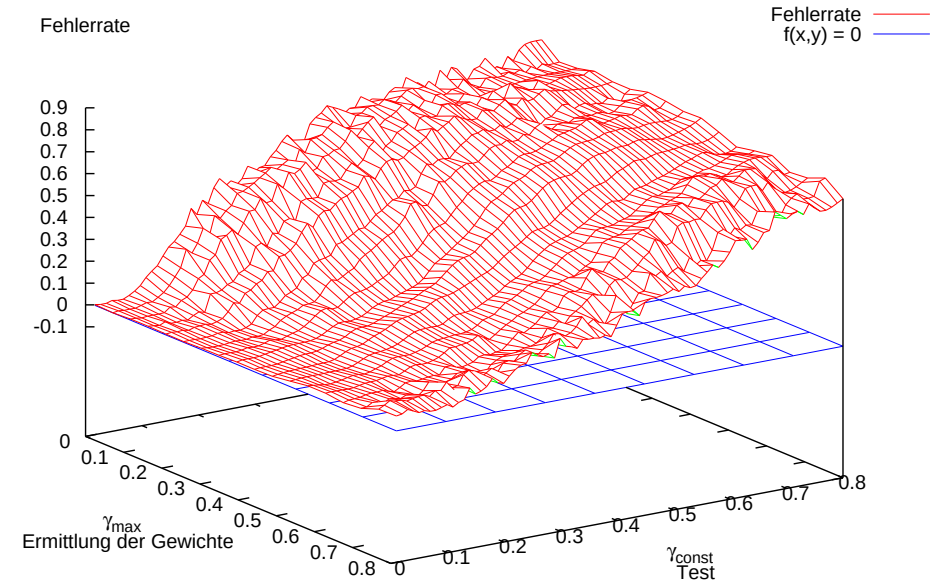


Abbildung 35: Fehlerrate in Abhängigkeit von der Verdeckung der Samples zur Gewichtsvektorermittlung und der Test-Samples.

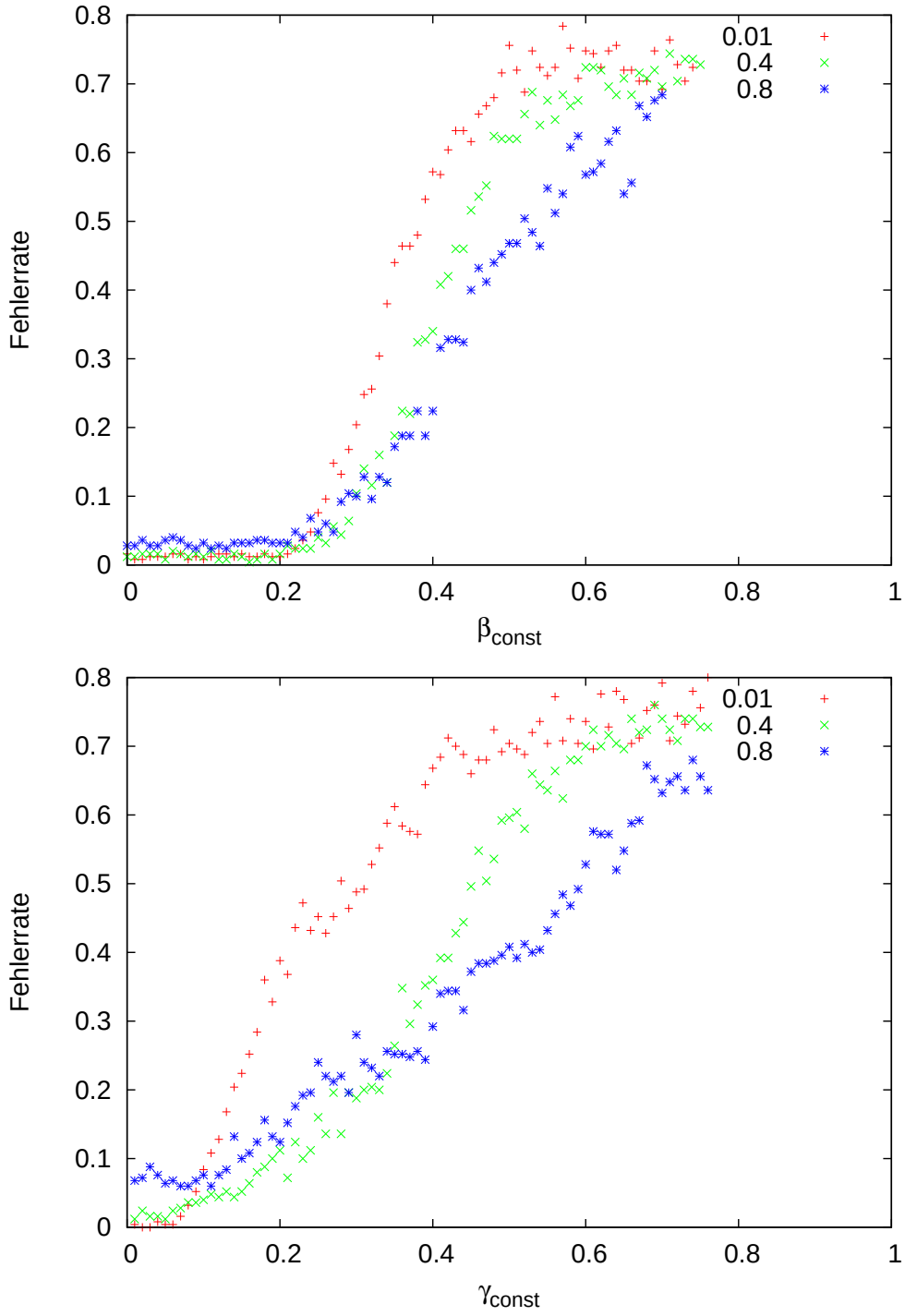


Abbildung 36: Verbesserung der Klassifikationsgenauigkeit mit der Erhöhung der oberen Schranke für die Verdeckungsrate bei der Ermittlung des Gewichtsvektors; Darstellung für $\beta^w_{max} = 0.01, 0.4, 0.8$ (oben); $\gamma^w_{max} = 0.01, 0.4, 0.8$ (unten);

Endentscheidung bekommen, als diejenigen, die für einen hohen Verdeckungsgrad trainiert sind. Die Abbildung 37 zeigt, dass bei einer weiteren Erhöhung von β^w_{max} bzw. γ^w_{max} von 0.8 auf 0.85 die Fehlerrate nicht weiter sinkt.

In Abbildung 38 ist ein Vergleich der Fehlerraten von AOC Klassifikator-Ensemble und einem einzelnen LCC-Klassifikator zu sehen. Hierbei wird der LCC-Klassifikator mit den Daten aller hierarchischen Ebenen trainiert, die im Ensemble unter den Mitgliedern verteilt werden. Diese Abbildung stellt die Abhängigkeit zwischen der Verdeckung γ^t_{const} an der X-Achse und der Klassifikationsfehlerrate an der Y-Achse dar. Ganz deutlich zu sehen ist die Verbesserung, die durch den Einsatz des Klassifikator-Ensembles zustande kommt. Annähernd auf dem ganzen Definitionsbereich liegen die Fehlerwerte von AOC unter denen eines einzelnen LCC-Klassifikator. Bei einer Verdeckung der Testdaten nah an $\gamma^t_{const} = 0.8$ ist eine eindeutige Zuordnung kaum mehr möglich, so dass die Klassifikationsgüte ungefähr dem zufälligen Raten entspricht.

Bei der Klassifikation 500 teilweise verdeckter Konturen mit $\gamma^t_{max} = 0.5$ hat sich mit dem Klassifikator-Ensemble der durchschnittliche Fehler von ca. 17 Prozent ergeben. Mit einem einzigen LCC-Klassifikator wurde mit dem gleichen Parameter-Satz der durchschnittliche Fehler von ca. 37 Prozent erzielt. Dies zeigt eine Verbesserung der Klassifikationsgüte bei dem Einsatz des Ensembles um etwa 20 Prozentpunkte. Der Durchschnitt wurde über die Daten von zwanzig Versuchen gebildet.

6.5 Adaboost Multi-Class

In diesem Abschnitt wird der Beitrag von Boosting auf die Klassifikationsgüte dargelegt. Es wird dafür die AdaBoost-Erweiterung, der Algorithmus *SAMME*, verwendet. Geboostet werden die einzelnen Ebenen des Adaptive Occlusion Classifiers. Statt eines Klassifikators werden jeweils $M \in \underline{10}$ Klassifikatoren pro Ebene eingesetzt. Für das Training und die Ermittlung der Gewichte werden zwei Mengen von jeweils 250 Samples verwendet. Die Testmenge besteht aus 100 Samples. Die Verdeckungsparameter sind gegeben mit $\beta^w_{max} = \beta^t_{max} = 0.5$. Als Konturmerkmal verwendet man hier den 14-dimensionalen Vektor der Fourier-Deskriptoren.

In Abbildung 40 (Anhang B) ist an der X-Achse die Anzahl der Iterationen M eingetragen. Die Y-Achse gibt die Fehlerrate an. Für jedes $M \in \underline{10}$ wird der Mittelwert über 200 Testdurchläufe gebildet. Die Standardabweichung ist mit Fehlerbalken dargestellt. In

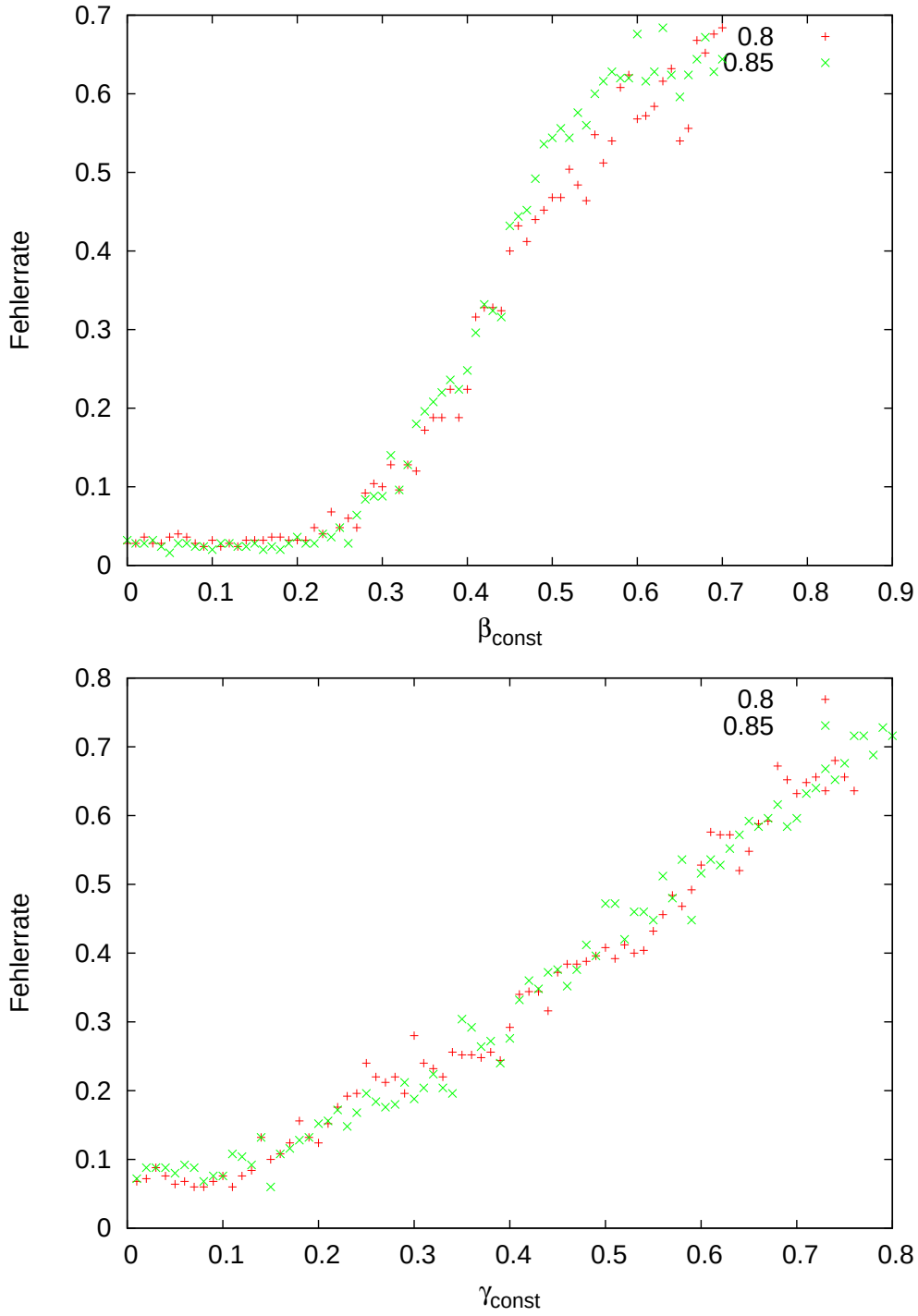


Abbildung 37: Keine Verbesserung der Klassifikationsgenauigkeit mit der Erhöhung von β^w_{max} bzw. γ^w_{max} von 0.8 auf 0.85; $\beta^w_{max} = 0.8, 0.85$ (oben) und $\gamma^w_{max} = 0.8, 0.85$ (unten);

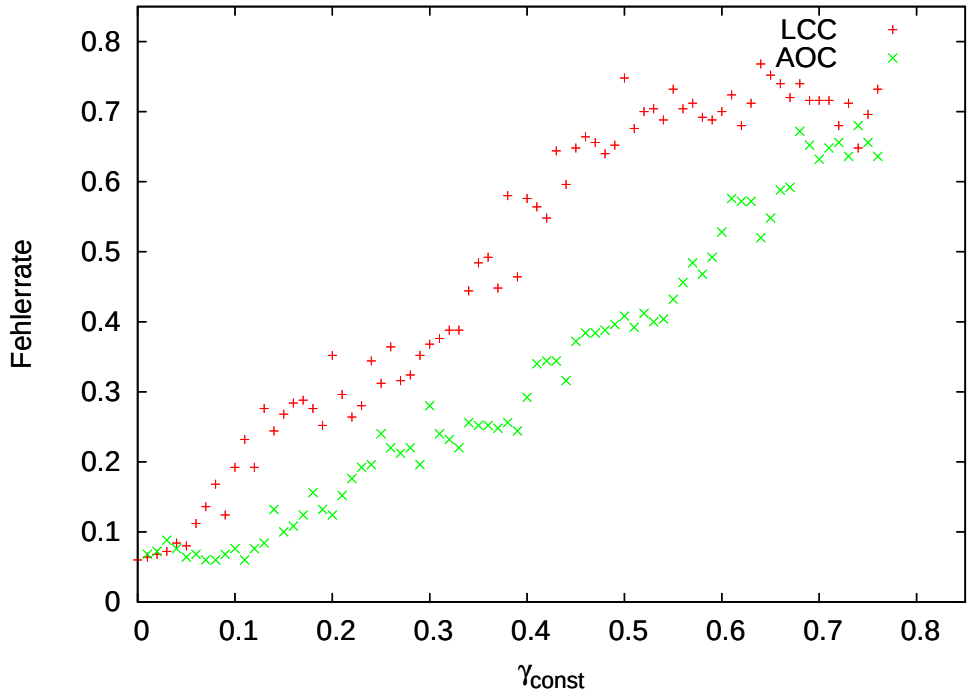


Abbildung 38: Vergleich der Klassifikationsergebnisse vom Klassifikator-Ensemble AOC (grün) und einem LCC-Klassifikator (rot). Die Werte von AOC für $\gamma^w_{max} = 0.8$ sind aus Abbildung 35 entnommen.

dieser Abbildung kann man gut erkennen, dass im Rahmen des Experiments die gemittelte Fehlerrate mit einer Erhöhung der Anzahl von Iterationen sich nicht stark verändert. Sie liegt im Intervall von 14 bis 16 Prozent.

6.6 Training und Klassifikation mit Raytrace-Objekten

Bei der Generierung künstlicher Bilder für diese Versuchsreihe (siehe Anhang C) orientierte man sich an den realen Figurmaßen, an dem existierenden experimentellen Setup und den realen Lichtverhältnissen. Der Abstand zwischen der Kamera und der Arbeitsfläche wurde in POV-Ray für die Kamerapositionierung übernommen. Für die Beleuchtung wurden vier Lichtquellen verwendet, um die Lichtverhältnisse des Projektaufbaus zu reproduzieren.

Die Bilder im POV-Ray wurden mit Hilfe der Translations- und Rotationsvorgaben generiert. Die Translation von links nach rechts, von oben nach unten, von links oben nach rechts unten und letztendlich von rechts oben nach links unten, ausgehend vom Zentrum

	R-R-S	R-S-R	R-S-S	S-R-R	S-R-S	S-S-R	S-S-S
Fehlerrate (Menge 1)	0.28	0.20	0.22	0.21	0.18	0.24	0.15
Fehlerrate (Menge 2)	0.27	0.19	0.24	0.22	0.21	0.23	0.22

Tabelle 1: Vergeich von Fehlerraten für unterschiedliche synthetische Daten

der Bildfläche, wurde mit jeweils 20 Schritten für die Erzeugung der Bildsequenzen durchgeführt. Zusätzlich wurden die Figuren in jedem Schritt, ausgehend von ihrer vorherigen Orientierung, um 2 Grad rotiert. Der gesamte Bilder-Pool besteht aus zwei Teilmengen, von denen jede aus 800 Objekten zusammengesetzt ist. Der Unterschied zwischen den Mengen besteht darin, dass die Translation jedes einzelnen Objektes in der zweiten Menge doppel so groß im Bezug zur Kamera-Position ist, wie die Translation der Objekte in der ersten Menge. Die mit den synthetischen Daten durchgeführten Experimente für eine steigende Anzahl der Training-Samples, unterschiedlichen Verdeckungsgrade und Merkmalsextraktionsverfahren haben im Vergleich zu den Tests mit echten Daten große Ähnlichkeiten gezeigt.

Hier werden die Testergebnisse vorgestellt, die durch die Aufnahme künstlich generierter Objekte in das Test- und Trainingsvorgang erhalten wurden. Innerhalb der Versuche wird das Klassifikator-Ensemble stets mit 350 Samples trainiert, 250 Konturen mit $\gamma^w_{max} = 0.5$ werden zur Ermittlung des Gewichtsvektors verwendet. Getestet werden 100 Samples mit Flächen-Verdeckung $\gamma^t_{max} = 0.5$. In der Tabelle 1 sind die Klassifikationsergebnisse, gemittelt über 10 Durchläufe, präsentiert. R (real) wird als Abkürzung für die Verwendung der Kamerabilder benutzt, S (synthetic) bezeichnet die Verwendung der POV-Ray Bilder. Die jeweiligen Spaltenbezeichnungen haben das Format R/S - R/S - R/S. Dieses spezifiziert, welche Art von Daten im Trainings-, Gewichtsermittlungs-, und Testvorgang verwendet wird. Beispielsweise steht "R-R-S" für den Einsatz von Kameradaten für das Training und die Ermittlung des Gewichtsvektors und für den Einsatz von POV-Ray Bildern für den Test des Klassifikator-Ensembles.

Führt man den Versuch nur mit Kamera-Bildern durch (R-R-R), so beträgt der durchschnittliche Fehler ca. 17 Prozent. Die Zeilen der Tabelle 1 fassen die Ergebnisse der Versuche mit den beiden oben beschriebenen Teilmengen des Daten-Pools zusammen. Die erste Zeile gibt die Fehlerraten an, die sich aus den Tests mit der ersten Teilmenge ergeben. Die zweite Zeile enthält die Ergebnisse für die Teilmenge mit der doppelten Translation.

Anhand der letzten Tabellenspalte (S-S-S) kann man erkennen, dass die Daten mit dem größeren Grad an perspektivischer Verzerrung aus der zweiten Menge deutlich schlechter klassifiziert werden können als die Daten aus der ersten Menge. Der Unterschied beträgt etwa 7 Prozent. Die Fehlerraten in den restlichen Tabellenspalten unterscheiden sich nicht erheblich für die beiden Teilmengen. In der vorletzten Spalte (S-S-R) sind die Testergebnisse mit echten Daten dargestellt, wobei das Training und die Ermittlung der Gewichte mit synthetischen Daten durchgeführt sind. Für die zweite Menge ist das Resultat ca. 6 Prozent schlechter im Vergleich dazu, wenn das Klassifikator-Training und die Gewichte-Ermittung mit echten Daten durchgeführt ist. Damit wird veranschaulicht, dass eine Automatisierung des Trainingsvorgangs durch die Verwendung synthetischer Daten möglich ist. Eine detaillierte Untersuchung in diesem Bereich könnte eine Fortführung dieser Arbeit ausmachen.

7 Zusammenfassung und Ausblick

In dieser Arbeit wurde gezeigt, dass unter den Rahmenbedingungen vom COSPAL-Projekt das *Adaptive Occlusion Classifier Ensemble* die Erkennung von Konturen teilweise verdeckter Objekte ermöglicht. Es wurde ermittelt, dass die Merkmalsvektoren, gewonnen aus Fourier-Deskriptoren, Zernike- und pseudo Zernike-Momenten, eine vergleichbare Klassifikationsgüte erzielen. Für eine randomisierte Kontur-Verdeckung mit $\beta^t_{max} = 0.5$ beträgt der Klassifikationsfehler für die 8-dimensionalen Verktoren durchschnittlich ca. 15 Prozent. Für eine randomisierte Flächen-Verdeckung mit $\gamma^t_{max} = 0.3$ beträgt der durschnittliche Klassifikationsfehler ca. 10 Prozent für die 12-dimensionalen Merkmalsvektoren. Eine Versuchsreihe zur Klassifikationsrobustheit beim variierenden Verdeckungsgrad zeigt eine deutliche Verbesserung der Erkennungsrate durch den Einsatz des Ensembles anstelle eines einzelnen LCC-Klassifikators. Mit POV-Ray synthetisch erstellte Objektbilder wurden in einer Reihe der Experimente in das Klassifikator-Training und -Testen integriert. Hierbei wurde das Potential einer Klassifikator-Training-Automatisierung durch das Ersetzen echter Kameraaufnahmen mit den künstlich erzeugten Bildern untersucht. Eine Erweiterung dieser Arbeit könnte darin bestehen, die hier erzielten Ergebnisse zu verbessern.

Die Art der Verdeckungen innerhalb dieser Arbeit ist auf eine lineare Verdeckung eingeschränkt. Eine Fortführung dieser Arbeit könnte sich mit nicht-linearen Verdeckungen beschäftigen. Eine andere Annahme besteht darin, dass die Verdeckung eines Objektes in einer einzigen geschlossenen Kontur resultiert. Dies kann im Allgemeinen nur für die konvexen Formen angenommen werden. In einer Erweiterung sollte berücksichtigt werden, dass sich nach einer Verdeckung eine Gruppe geschlossener Konturen ergeben kann. Die Figuren dürfen innerhalb der Versuche im COSPAL-Projekt nicht auf der Seite liegen. Hierbei wird angenommen, dass durch die zur Arbeitsfläche parallele Bewegung der Kamera innerhalb eines eingeschränkten Bereiches nur geringe perspektivische Konturverzerrungen entstehen. Eine Erweiterung der Arbeit könnte sich damit beschäftigen, Figuren, aufgenommen unter einem beliebigen Blickwinkel und bei einer beliebigen Positionierung, zu klassifizieren. Zur Vereinfachung des Training-Prozesses könnten hierfür POV-Ray-Bilder verwendet werden. In einer weiteren Verallgemeinerung des Experimentes könnte eine größere Anzahl von Figurklassen verwendet werden. In einer Fortsetzung dieser Arbeit könnten, aufbauend auf Kontur- und Flächen-Verdeckungsverfahren, Versuche mit unvollständigen Konturen durchgeführt werden.

A Subsampling

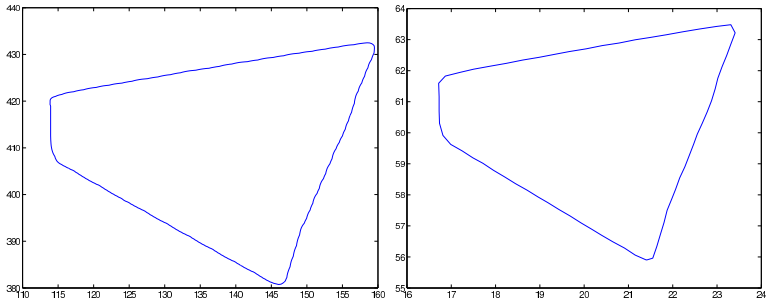


Abbildung 39: links: Originalkontur mit $L = 350$; rechts: Kontur nach Subsampling, hier beträgt die Anzahl der diskreten Punkte 64.

B AdaBoost

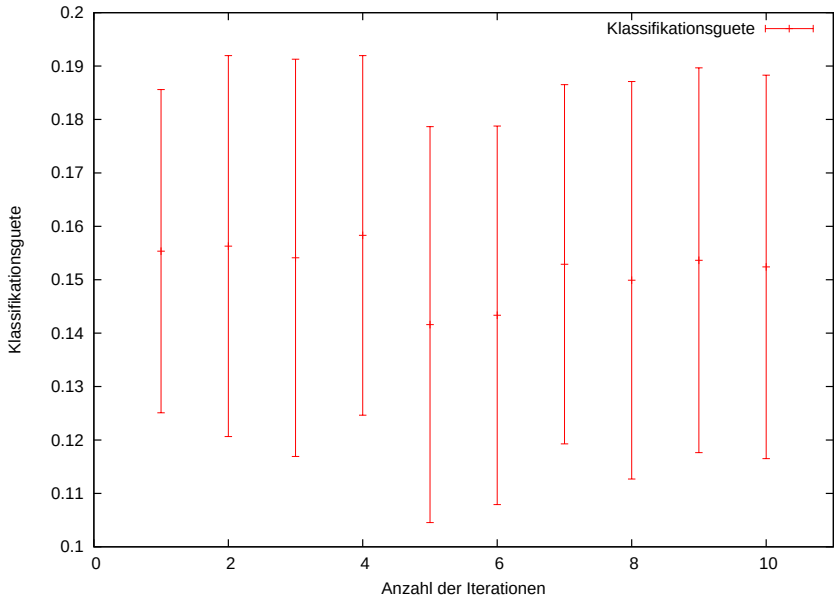


Abbildung 40: Klassifikationsfehlerrate in Abhängigkeit von Anzahl M der pro Ebene verwendeten Klassifikatoren; Mittelwert gebildet über 200 Durchläufe; Fehlerbalken 1σ.

C POV-Ray Figuren

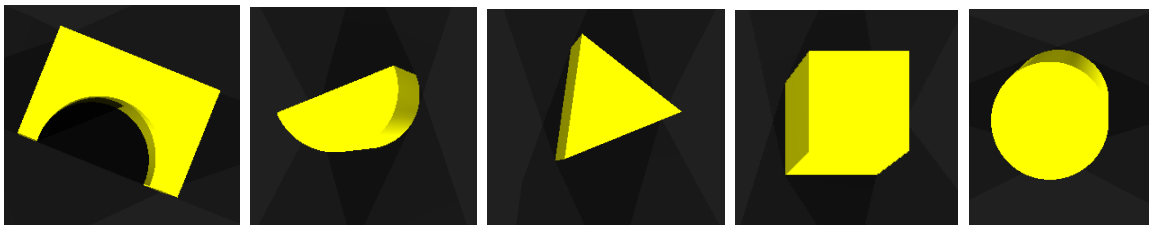


Abbildung 41: Einige Beispiele vom POV-Ray Figuren.

Literatur

[1] FFTW (http://www.fftw.org).

[2] Persistence of vision raytracer documentation (http://www.povray.org/documentation/).

[3] Persistence of vision raytracer (http://www.povray.org/).

[4] Gholamreza Amayeh, Ali Erol, George Bebis, and Mircea Nicolescu. Accurate and efficient computation of high order zernike moments. In *ISVC*, pages 462–469, 2005.

[5] Yannis Avrithis, Yiannis Xirouxakis, and Stefanos Kollias. Affine-invariant curve normalization for object shape representation, classification and retrieval. *Machine Vision and Applications*, 13(2):80–94, 2001.

[6] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[7] Eamon B. Barrett, Paul M. Payton, Nils N. Haag, and Michael H. Brill. General methods for determining projective invariants in imagery. *CVGIP: Image Underst.*, 53(1):46–65, 1991.

[8] Christopher Brown. Numerical evaluation of differential and semi-differential invariants. In Joseph L. Mundy and Andrew Zisserman, editors, *Geometric invariance in computer vision*, chapter 10, pages 215–227. MIT Press, Cambridge, MA, USA, 1992.

[9] Horst Bunke. Hybrid methods in pattern recognition. In *Proc. of the NATO Advanced Study Institute on Pattern recognition theory and applications*, pages 367–382, London, UK, 1987. Springer-Verlag.

[10] Tilman Butz. *Fourier transformation for pedestrians*. Springer, Berlin, 2006.

[11] James W. Cooley and John W. Turkey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[12] Thomas G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000.

[13] Sherif Sami El-Dabi, Refat Ramsis, and Aladin Kuwait. Arabic character recognition system: a statistical approach for recognizing cursive typewritten text. *Pattern Recogn.*, 23(5):485–495, 1990.

[14] Yoav Freund and Robert E. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[15] Yoav Freund and Robert E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, September 1999.

[16] Matteo Frigo and Steven G Johnson. FFTW:an adaptive software architecture for the FFTW. In *Proc. IEEE International Conference on Acoustics, SPeech, and Signal Processing*, volume 3, pages 1381–1383, 1998.

[17] Anarta Ghosh. *Robustness of Shape Descriptors and Dynamics of Learning Vector Quantization*. PhD thesis, Rijksuniversiteit Groningen, 2007.

[18] Luc J. Van Gool, Michael H. Brill, Eamon B. Barrett, Theo Moons, and Eric Pauwels. Numerical evaluation of differential and semi-differential invariants. In Joseph L. Mundy and Andrew Zisserman, editors, *Geometric invariance in computer vision*, chapter 15, pages 293–309. MIT Press, Cambridge, MA, USA, 1992.

[19] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(10):993–1001, 1990.

[20] John Howie. *Complex Analysis*. Springer-Verlag, 2003.

[21] A. Khotanzad and Y. H. Hong. Invariant image recognition by zernike moments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(5):489–497, 1990.

[22] Volodymyr Kindratenko. *Development and Application of Image Analysis Techniques for Identification and Classification of Microscopic Particles*. PhD thesis, University of Antwerpen, 1997.

[23] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[24] Kah Bin Lim, Tiehua Du, and Hao Zheng. 2d partially occluded object recognition using curve moments. In *Proceedings of the Seventh IASTED International Conference Computer Graphics And Imaging*, Kauai, Hawaii, USA, August 17-19 2004.

[25] Joseph L. Mundy and Andrew Zisserman, editors. *Geometric invariance in computer vision*. MIT Press, Cambridge, MA, USA, 1992.

[26] Joseph L. Mundy, Andrew Zisserman, and David A. Forsyth, editors. *Applications of Invariance in Computer Vision, Second Joint European - US Workshop, Ponta Delgada, Azores, Portugal, October 9-14, 1993, Proceedings*, volume 825 of *Lecture Notes in Computer Science*. Springer, 1994.

[27] Andreas Opelt, Axel Pinz, and Andrew Zisserman. A boundary-fragment-model for object detection. In *ECCV (2)*, pages 575–588, 2006.

[28] H. Prehn and G. Sommer. Incremental classifier based on a local credibility criterion. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, AIA 2007, February 12-14, Innsbruck, Austria*, pages 372–377. ACTA Press, 2007.

[29] Parichart Putjarupong, Chuchart Pintavirooj, Withawat Withayachumnankul, and Manas Sangworasil. Image registration exploiting five-point coplanar perspective invariant and maximum-curvature point. In *Journal WSCG*, volume 12, pages 341–348, 2004.

[30] Thomas H. Reiss. *Recognizing Planar Objects Using Invariant Image Features*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1993.

[31] Ehud Rivlin and Isaac Weiss. Local invariants for recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(3):226–238, 1995.

[32] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering, 2007.

[33] Mirela Tanase, Remco C. Veltkamp, and Herman J. Haverkort. Multiple polyline to polygon matching. In *ISAAC*, pages 60–70, 2005.

[34] Cho-Huak Teh and Roland T. Chin. On image analysis by the methods of moments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(4):496–513, 1988.

[35] Isaac Weiss. Projective invariants of shapes. In *Proc. DARPA Image Understanding Workshop*, pages 1125–1134, 1988.

[36] Isaac Weiss. Noise-resistant invariants of curves. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(9):943–948, 1993.

[37] Isaac Weiss and Manjit Ray. Recognizing articulated objects using a region-based invariant transform. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1660–1665, 2005.

[38] Emil Wolf and A.B. Bhatia. On the circle polynomials of zernike and related orthogonal sets. In *Proc. Cambridge Phil. Soc.*, volume 50, pages 40–48, 1954.

[39] Frederik Zernike. Beugungstheorie des schneidenverfahrens und seiner verbesserten form, der phasenkontrastmethode (diffraction theory of the cut procedure and its improved form, the phase contrast method). *Physica*, 1:689–704, 1934.

[40] Ji Zhu, Saharon Rosset, Hui Zou, and Trevor Hastie. Multi-class adaboost, January 2006.

Danksagung

Zuerst möchte ich Herward Prehn und Prof. Dr. Sommer für die großartige Hilfe und die Wegweisung danken. Ebenso möchte ich den weiteren Mitgliedern des Lehrstuhls und insbesondere den Administratoren Gerd Diesner und Henrik Schmidt für die Beantwortung vieler Fragen danken. Für das Stipendium, das mir erlaubt hat mich völlig meiner Arbeit zu widmen, möchte ich mich bei der DAB Gruppe Kiel und insbesondere bei Annelore Brammer bedanken. Herzlichst möchte ich mich bei dem Mitarbeiter der mathematischen Seminarbibliothek Lennart Giesse für die Hilfe mit den Literaturbestellungen bedanken.

Meinen Büronachbarn Christopher Friedt und Marco Chavarria danke ich dafür, dass sie mir zur späten Stunde und am Wochenende, im Büro und auch beim Kaffee-Trinken häufig Gesellschaft geleistet haben. Ich danke meinen Eltern und meinem Freund Onno für die außerordentliche Unterstützung. Für das Korrekturlesen bedanke ich mich bei Florian Hoppe und Wolfgang Hildebrandt.

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfaßt und ausschließlich die angegebenen Hilfsmittel und Quellen verwendet habe.

Kiel, den