ALBERT-LUDWIGS-UNIVERSITÄT
FREIBURG
INSTITUT FÜR INFORMATIK

Lehrstuhl für Mustererkennung und Bildverarbeitung
Prof. Dr. Hans Burkhardt

Automatische Selektion von optimalen Kernfunktionen für Grauwertbasierte Invarianten

Diplomarbeit

Thorsten Schmidt

Oktober 2004 – April 2005

Erklärung

Hiermit erkläre ich, daß die vorliegende Arbeit von mir selbständig und nur unter Verwendung der aufgeführten Hilfsmittel erstellt wurde.

Freiburg, den

Zusammenfassung

Ein wichtiger Schritt in der Mustererkennung ist die Extraktion invarianter Merkmale aus gegebenen Daten. Dies erfordert oft genaue Kenntnis der Problemstellung und Experten, die die Extraktion für das spezifische Problem anleiten. In dieser Arbeit soll ein Verfahren erläutert werden, das selbständig in der Lage ist für beliebige gegebene Datenbeispiele basierend auf der Gray Value Cooccurrence Distribution diskriminative Kernfunktionen zu bestimmen, die eine möglichst gute Klassifikation der gegebenen, wie auch weiterer Daten ermöglichen. Die Kernfunktionen sollen dabei so beschaffen sein, dass ihre Anwendung auf weitere Daten möglichst schnell ausgeführt werden kann. Die theoretischen Grundlagen des Verfahrens werden im folgenden erläutert, sowie deren algorithmische Umsetzung. Es wurden praktische Versuche an realen Daten aus der Biologie unternommen, deren Resultate am Ende dieser Arbeit präsentiert werden.

Inhaltsverzeichnis

Verwendete Begriffe

Akronyme

CLSM Confocal Laser Scanning Microscope

DFT Discrete Fourier Transform (Diskrete Fourier Transformation)

FFT Fast Fourier Transform (Schnelle implementation der diskreten Fourier-Transformation)

GLCM Gray Level Cooccurrence Matrix

GSI Gray Scale Invariant

GVCD Gray Value Cooccurrence Distribution

IGLCM Isotropic Gray Level Cooccurrence Matrix (siehe auch GLCM)

KLT Karhunen Loève Transformation, Hauptachsentransformation

PCA Principal Component Analysis (= KLT)

SSI Scale Space Invariant

SVM Supportvektormaschine

Mathematische Notation

$\mathcal{F}\{.\}$	Diskrete Fourier-Transformation
$\mathcal{F}^{-1}\{.\}$	inverse Fourier-Transformation
$(f * g)(t)$	Faltung von f und g: $\left(f * g\right)(t) = \int_{-\infty}^{\infty} f(\tau) \cdot g(t-\tau)\ \mathrm{d}\tau$
$(f \otimes g)(t)$	Korrelation von f und g: $\left(f \otimes g\right)(t) = \int_{-\infty}^{\infty} f(\tau) \cdot g(t+\tau)\ \mathrm{d}\tau$
$\|.\|$	L_2-Norm (euklidische Norm)
$\delta(x)$	Dirac-Impuls (Definition siehe Seite 11 Gleichung 3.1)

1. Einführung

1.1. Kernselektion in der Mustererkennungskette

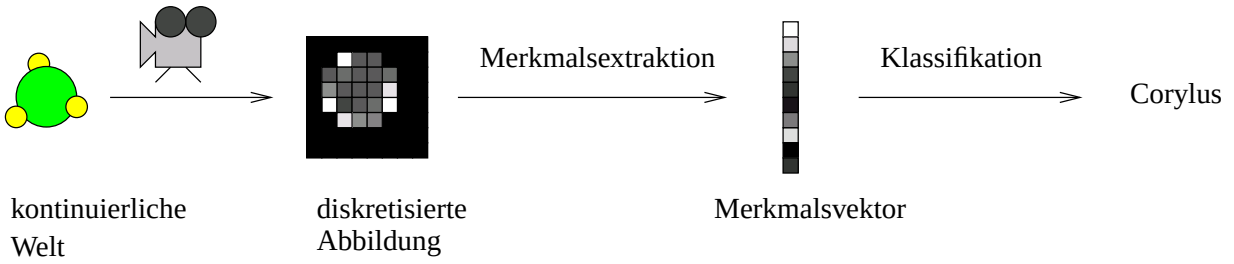


Abbildung 1.1.: Die Mustererkennungskette

Die Selektion einer Kernfunktion ergibt eine Vorschrift aus den digitalisierten Daten einen Merkmalsvektor zu berechnen. Daher fällt sie in den Bereich der Merkmalsextraktion. Sind die Kernfunktionen bereits bekannt, gilt dies auch und die Kette wird wie abgebildet abgearbeitet. Meist werden Kernfunktionen mit großem Aufwand und einigem Vorwissen über das Problem von Hand konstruiert um möglichst diskriminative Merkmale extrahieren zu können. In dem im Verlauf der Arbeit vorgestellten Verfahren zur Bestimmung von Kernfunktionen, ist dieses Wissen nicht erforderlich, allerdings eine gewisse Menge an Beispieldaten, deren Klassenzugehörigkeit bekannt ist, anhand derer über statistische Betrachtungen diskriminative Merkmale extrahiert werden.

1.2. Die Gray Value Cooccurrence Distribution

Definition 1.2.1: *Gray Value Cooccurrence Distribution (GVCD)*
Die Gray Value Cooccurrence Distribution ist eine Verteilungsdichtefunktion:

$$\begin{aligned} \mathrm{gvcd}_r : \mathbb{R}^2 &\rightarrow [0,1] \\ \mathrm{gvcd}_r\,(v_1, v_2) &\mapsto p\,(\text{Zwei Punkte mit Abstand } r \text{ haben Grauwerte } v_1 \text{ und } v_2) \end{aligned}$$

Obige Definition ist eng angelehnt an die Definition der isotropen Gray Level Cooccurrence Matrix (IGLCM) [WJL97]. Die Idee ist sogar noch etwas älter und stammt von R. Haralick, der den Begriff der Gray Level Cooccurrence Matrix 1973 eingeführt hat [HSD73]. In der Originaldefinition wurden jedoch nur diskrete Graustufen (gray levels) unterschieden, und aus Gründen der Berechenbarkeit betrachten alle mir bekannten Algorithmen zur Berechnung der GLCM nur eine sehr kleine Anzahl von mit Abstand r benachbarten Punkten jedes Signalpunktes. Diese Einschränkungen sind nun für die Berechnung der kontinuierlichen GVCD nicht mehr in dieser Form zulässig. Natürlich ist die Berechnung der vollständigen Verbundverteilung, die durch die GVCD dargestellt wird nicht möglich, worauf in Abschnitt 3.1 im Detail eingegangen wird. Nichts desto trotz ist obige Definition für theoretische Betrachtungen sehr wertvoll, da sie in dieser Form als einfache Verteilungsfunktion die Nutzung bekannter Verfahren und Ideen aus der Stochastik ermöglicht.

1.3. 2-Punkt-Invarianten

Das Ziel dieser Arbeit ist die Suche nach möglichst optimalen Kernfunktionen für 2-Punkt-Invarianten. Um den Zusammenhang zwischen diesen Invarianten und der Gray Value Cooccurrence Distribution zu verstehen, möchte ich kurz auf die 2-Punkt-Invarianten, wie sie in ihrer ursprünglichen Form 2002 von Olaf Ronneberger [RBS02] veröffentlicht wurden, eingehen. Grundsätzlich werden alle im folgenden ausgeführten Betrachtungen auf 2D-Signale bezogen. Dies dient nur der Vereinfachung und dem besseren Verständnis. Analoge Betrachtungen sind ebenso in 3 Dimensionen möglich.

Eine Klasse von Verfahren zur Berechnung von Invarianten bezüglich einer Transformationsgruppe sind die Gruppenmittel [SM94]. Ein bekannter Vertreter dieser Klasse sind die Haar-Integrale. Ihr Prinzip ist recht einfach und garantiert invariante Merkmale bezüglich der betrachteten Gruppe G durch Integration über alle Gruppenelemente $g \in G$. Um neben der Invarianz auch Separierfähigkeit zu erreichen, wird zur Integration eine nichtlineare Kernfunktion $f\left(\mathbf{X}\right)$ hinzugezogen, die auf jede Transformation des Signals angewendet wird. In Formeln beschrieben ist das Haar-Integral definiert durch:

$$T[f]\left(\mathbf{X}\right) = \int_G f\left(g\mathbf{X}\right) \mathrm{d}g \tag{1.1}$$

Dabei ist g ein Gruppenelement (z.B. eine Rotation um einen bestimmten Winkel oder eine Translation einer bestimmten Länge und Richtung) und $\mathbf{X}$ das Signal. Die nichtlineare Kernfunktion $f\left(\mathbf{X}\right)$ ist von besonderer Bedeutung. Klassische Kernfunktionen sind Monome, die eine gewisse Nachbarschaft um den betrachteten Punkt hinzuziehen um lokale Eigenschaften des Signals zu beschreiben. Neben der notwendigen Invarianz ist eine spezielle Eigenschaft der Haar-Integrale ihre Robustheit gegen kleine lokale Veränderungen des Signals, wie Deformationen oder Überlappungen durch das Signal beschriebener Objekte. Dies erklärt sich dadurch, dass durch den kleinen Einflussbereich der Kernfunktion nur wenige einzelne Kernauswertungen ihren Wert verändern, was sich im Mittel nur noch minimal auswirkt. Der Nachteil der Haar-Integrale besteht in der sehr aufwendigen Berechnung, da im Allgemeinen die Integration über alle Gruppenelemente und die punktweise Auswertung der Kernfunktion explizit berechnet werden muss. Eine Möglichkeit den Berechnungsaufwand gering zu halten, ist die Schätzung der Invarianten mit Hilfe einer Monte Carlo Integration [SS99]. Diese Idee wird im weiteren Verlauf zur Berechnung der GVCD wieder aufgegriffen werden.

Betrachtet man Kernfunktionen vom Typ $f\left(\mathbf{X}\left(\vec{x}_1\right), \mathbf{X}\left(\vec{x}_2\right), \mathbf{X}\left(\vec{x}_3\right), \ldots\right)$ ist es möglich anstelle des gesamten Signals, nur die von der Kernfunktion betrachteten Punkte zu transformieren. Dies reduziert den Aufwand erheblich und führt zu folgender Neuformulierung:

$$\left(g\mathbf{X}\right)\left(\vec{x}\right) = \mathbf{X}\left(s_g\left(\vec{x}\right)\right) \tag{1.2}$$

Hier bezeichnet s_g die Transformation der Punkte der Kernfunktion mit Gruppenparameter g.

Ein Spezialfall der Haar-Integrale, der die gerade erwähnte Eigenschaft besitzt, sind die 2-Punkt-Grauwert-Invarianten. Sie zeichnen sich durch die Wahl einer besonders einfachen Kernfunktion aus, die wie der Name bereits aussagt, nur zwei Punkte des Signals betrachtet. Ihre allgemeine Form ist:

$$f\left(\mathbf{X}\right) = f\left(\mathbf{X}\left(\vec{0}\right), \mathbf{X}\left(\vec{q}\right)\right) \tag{1.3}$$

Ihr einfachster Vertreter ist die multiplikative Verknüpfung der beiden Punkte, wie in Gleichung 1.4 dargestellt.

$$f\left(\mathbf{X}\right) = \mathbf{X}\left(\vec{0}\right) \cdot \mathbf{X}\left(\vec{q}\right) \tag{1.4}$$

Will man nun eine 2-Punkt-Invariante für die Gruppe der euklidischen Bewegungen berechnen, lässt sich mit Hilfe dieser einfachen Kernfunktion die schnelle Fourier-Transformation verwenden. Die theoretischen Grundlagen und die Herleitung sind in [RBS02] nachzulesen.

Sei $\mathbf{C}_{\|\vec{q}\|}$ ein Kreis mit Radius $\|\vec{q}\|$ und $\vec{x} = (x_1, x_2)^T$ eine Position innerhalb des 2D-Signals, dann lässt sich die 2-Punkt-Invariante mit Kernfunktion f definiert wie in Gleichung 1.4 folgendermaßen berechnen:

$$T[f](\mathbf{X}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{\mathbf{X}}{\|\mathbf{X}\|}(\vec{x}) \cdot \left(\frac{\mathbf{X}}{\|\mathbf{X}\|} * \mathbf{C}_{\|\vec{q}\|}\right)(\vec{x})\ \mathrm{d}x_1\ \mathrm{d}x_2 \tag{1.5}$$

Wird für die Faltung die schnelle Implementation mit Hilfe der schnellen Fourier-Tranformation verwendet, ist die Laufzeit der gesamten Integration in $O(N \log N)$, mit N = #Abtastpunkte von $\mathbf{X}$.

Eine etwas andere, aber gleichwertige Berechnungsvorschrift ergibt sich aus der Formel:

$$T[f](\mathbf{X}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbf{C}_{\|\vec{q}\|}(\vec{x}) \cdot \frac{(\mathbf{X} \otimes \mathbf{X})}{\|\mathbf{X}\|^2}(\vec{x})\ \mathrm{d}x_1\ \mathrm{d}x_2 \tag{1.6}$$

Dies wird für die Berechnung der GVCDs im weiteren Verlauf dieser Arbeit noch von Bedeutung sein.
In beiden Beschreibungen wird das Signal $\mathbf{X}$ auf 1 normiert, um die Invarianten zu erhalten. Dies ist nicht zwingend notwendig und hat sogar für die Berechnung der GVCD im weiteren einen Nachteil, der an geeigneter Stelle wieder aufgegriffen wird.

Aus der bisherigen Beschreibung wird nun der Zusammenhang zwischen 2-Punkt-Invariante und GVCD noch nicht klar. Erst durch eine Umformulierung der allgemeinen Kernfunktion aus Gleichung 1.3 und einer Vertauschung der Anwendung der Gruppenoperation g und der Kernfunktion f in der Berechnungsformel für Haar-Integrale (1.1) wird dieser deutlich (vgl. [Ron04]).

Der erste Schritt ist die folgende Umformulierung der Kernfunktion aus Gleichung 1.3:

$$f(\mathbf{X}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta\left(v_1 - \mathbf{X}\left(\vec{0}\right)\right) \cdot \delta\left(v_2 - \mathbf{X}\left(\vec{q}\right)\right) \cdot f(v_1, v_2)\ \mathrm{d}v_1\ \mathrm{d}v_2 \tag{1.7}$$

Setzt man nun diese neu formulierte Kernfunktion in die Haar-Integral Berechnung ein und führt die erwähnte Vertauschung von f und g durch, erhält man:

$$T[f](\mathbf{X}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(v_1, v_2) \cdot \underbrace{\int_G \delta\left(v_1 - \mathbf{X}\left(s_g\left(\vec{0}\right)\right)\right) \cdot \delta\left(v_2 - \mathbf{X}\left(s_g\left(\vec{q}\right)\right)\right)\ \mathrm{d}g}_{\mathbf{gvcd}_{\|\vec{q}\|}}\ \mathrm{d}v_1\ \mathrm{d}v_2 \tag{1.8}$$

Betrachtet man obige Formel genau, erkennt man, dass die dort berechnete GVCD bis auf einen konstanten Faktor der Definition in Abschnitt 1.2 entspricht. Die verwendete Formulierung entspricht hier einem kontinuierlichen 2D Histogramm, in dem, anstelle der Wahrscheinlichkeit, die Häufigkeit des Auftretens jedes Grauwertpaares gezählt wird. Theoretisch sind beide Formulierungen für die Bestimmung einer Kernfunktion $f(v_1, v_2)$ gleichwertig. In dieser Arbeit wird die histogrammähnliche Variante jedoch aus Gründen der Rechengenauigkeit in den praktischen Algorithmen bevorzugt. Warum dies so ist, wird in Kapitel 3 näher begründet.

2. Datenbeschreibung

Für alle in dieser Arbeit ausgeführten Experimente, wurden zwei verschiedene Datenbanken verwendet:

LSM 3D Pollenaufnahmen :

Anzahl Datensätze: 385

Anzahl Klassen: 26

Datenformat: 3D Grauwert Voxel-Daten mit einer Auflösung von $128 \times 128 \times 128$ Voxel, 256 Graustufen gespeichert im netCDF Format

Beschreibung: Jeder Datensatz enthält die Aufnahme eines Pollen aus einer von 26 Pollensorten. Die volumetrischen Daten wurden mithilfe eines Konfokalen Laser Scanning Mikroskops aufgenommen, indem die fokale Ebene der abbildenden Optik schrittweise verändert wurde, um so verschiedene einzelne Ebenen der Objekte abzubilden. Dieses Verfahren wird in der Literatur als „Optical Sectioning Microscopy" bezeichnet (vgl. [RN00]). Die so entstandenen Bilder wurden anschließend wieder zu 3D Datensätzen zusammengefügt und in netCDF Dateien abgespeichert.
Für die in dieser Datenbank enthaltenen Aufnahmen wurde ein Öl-Immersions Objektiv mit 40facher Vergrößerung verwendet. Die Proben wurden für die Aufnahmen mit Wellenlängen von 450 - 490nm erregt und eine Emissionen ab einer Wellenlänge von 515nm wurde aufgenommen. Ein Voxel beschreibt ein kubisches Volumen mit einer Kantenlänge von ca. 200nm.

2-Kanal 2D Pollenaufnahmen :

Anzahl Datensätze: 1742

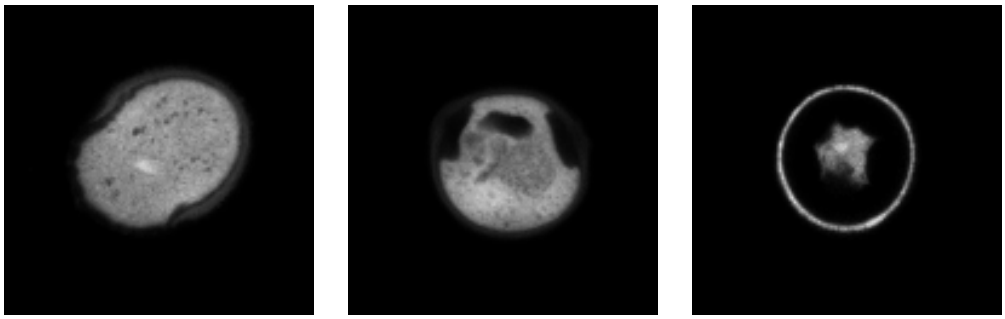
Anzahl Klassen: 12

Datenformat: 2D Grauwert Daten mit 256 Graustufen. Jeder Datensatz enthält zwei Bilder mit einer Auflösung von 512×576 Pixel nach Hinzufügen eines Pads. Das Dateiformat ist auch hier netCDF

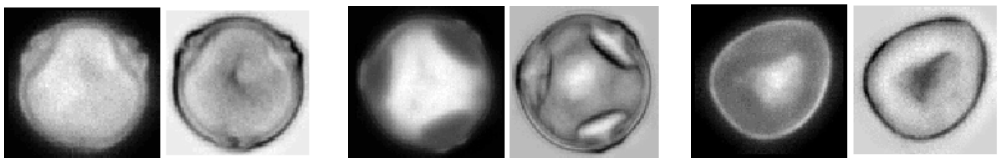
Beschreibung: Jeder Datensatz enthält 2 Aufnahmen eines Pollen aus einer von 12 Pollensorten. Eines der Bilder wurde unter Erregung des abgebildeten Pollen mit einer Fluoreszenzlampe aufgenommen, das andere bei normalem Lichtspektrum nach dem Durchlichtverfahren. Auch in diesem Fall wurde zunächst eine Reihe von Bildern verschiedener Fokusebenen aufgenommen. und anschließend das insgesamt schärfste in die Datenbank übertragen. Durch Verwendung eines konventionellen Mikroskops, weisen auf diese Art gewonnene Daten nur eine weitaus geringere Auflösung entlang der optischen Achse auf. Hinzu kommt, dass die durch die Optik fokussierte Ebene, durch Anteile aus den benachbarten Ebenen außerhalb des Fokus gestört werden. Dieser Effekt kann durch eine digitale Aufbereitung der Daten mit Hilfe von Entfaltungstechniken reduziert werden. Die hier gewonnenen Daten, entstammen jedoch einer Machbarkeitsstudie zur automatischen Klassifikation und Auszählung von Pollen, nach Möglichkeit in Echtzeit. Das in der autonomen Pollenzählstation eingebaute Mikroskop, ist nicht mit der Präzisionsoptik eines guten Labormikroskops ausgestattet, was für eine vernünftige Rekonstruktion notwendig ist. Daher wurde auch hier auf eine entsprechende Vorverarbeitung verzichtet wurde obwohl die Bilder dieser Datenbank auf einem Zeiss Labormikroskop aufgenommen wurden.

Für die Aufnahmen wurde ein Objektiv mit 20facher Vergrößerung und einer numerischen Apertur von 0.5 verwendet. Für den Fluoreszenzkanal wurden die Proben mit einer Wellenlänge von 395 - 440nm erregt und Wellenlängen ab 470nm aufgenommen. Ein Pixel beschreibt eine quadratische Fläche mit einer Kantenlänge von ca. 320nm.

Beispiele für die aufgenommenen Daten sind in Abb. 2.1 zu sehen.



Ahorn 3D | Hasel 3D | Eibe 3D



Birke 2D | Hasel 2D | Eibe 2D

Abbildung 2.1.: Testdatensätze für Einzelbetrachtungen im Verlauf der Arbeit: Die obere Zeile zeigt die mittlere Ebene des entsprechenden 3D-Datensatzes. In der unteren Zeile sieht man links die Fluoreszenz-Aufnahme und rechts die korrespondierende Durchlichtaufnahme

3. Diskrete Berechnung der GVCD

3.1. Annahmen

Das theoretische Konstrukt der Gray Value Cooccurrence Distribution geht von kontinuierlichen (mehrdimensionalen) Signalen aus. Uns stehen diese Daten jedoch in dieser Form nicht zur Verfügung. Wir haben in der Regel nur einen kleinen abgetasteten Ausschnitt des vollständigen Signals, der zudem noch in der Grauwertauflösung quantisiert ist. Damit treffen wir auf alle bekannten Probleme bei der Überführung der kontinuierlichen Welt in die digitale Signalverarbeitung. Es stellt sich die Frage, wie geht man mit dieser vergleichsweise dürftigen Approximation um?
Da diese Fragestellung eine der zentralen Fragestellungen der digitalen Bildverarbeitung ist, und sie in vielen gängigen Grundlagenbüchern (vgl. [Sch88]) ausgiebig behandelt wird, will ich hier nur die Annahmen, die für diese Arbeit getroffen wurden, näher erläutern:

Windowing Der endlichen Ausdehnung der Signale wird Rechnung getragen, indem die Signale mit Nullen gepadded und mit dem Pad als sich zyklisch wiederholend angenommen werden. Dies erhöht die Datenmenge natürlich erheblich, ermöglicht aber die Verwendung der schnellen Fourier-Transformation, was den erhöhten Speicheraufwand wieder rechtfertigt. Dies ist auch der Grund für die im letzten Kapitel erwähnte Bevorzugung der histogrammähnlichen GVCD, da durch das Pad mindestens drei Viertel (in 2D), bzw. sieben Achtel (in 3D) der Daten nur Nullen enthalten. Dadurch werden die Stellen der GVCD, die in einer Grauwertdimension die 0 betrachten astronomisch groß im Vergleich zu den restlichen, dabei enthalten sie fast nur Information über die ungefähre Größe des durch das Signal beschriebenen Objektes und kaum Information über die Textur. Das Integral über die GVCD wächst quadratisch mit der Anzahl Abtastpunkte, wodurch extrem kleine Normierungsfaktoren zu verwenden wären. Der auf realen Daten auftretende enorme Dynamikumfang der GVCD und dieser sehr kleine Faktor führen zu merklichen numerischen Ungenauigkeiten, die durch unterlassen der Normierung von vorneherein ausgeschlossen werden können.

Abtastung und Quantisierung Da nur diskrete Stützstellen des Signals zur Verfügung stehen, wurde mangels genauerer Kenntnis der zugrundeliegenden Funktion bei Berechnungen im Ortsbereich zwischen den Stützstellen linear interpoliert.
Bei Verwendung der Fourier-Transformation wird das Signal durch eine Summe von komplexen Exponentialfunktionen approximiert. Dies führt zu einer anderen, differenzierbaren Interpolation zwischen den Stützstellen, ändert jedoch durch den grundsätzlich verschiedenen Ansatz auch die Ergebnisse ein wenig im Vergleich zur linearen Interpolation.
Beiden Interpolationsverfahren gemeinsam ist, dass der quantisierte Grauwertbereich wieder kontinuierlich wird. Die minimalen und maximalen Grauwerte bleiben erhalten.

3.2. Praktische Vereinfachungen

Neben diesen allgemein bekannten Problemen hält die Berechnung der GVCD noch eine Reihe weiterer Herausforderungen bereit, die nur durch eine Entspannung der Problemstellung durch eine Reihe von Vereinfachungen zu begegnen sind:

Beschreibung der Funktion Die GVCD ist eine kontinuierliche Funktion über alle Grauwertpaare. Diese ist jedoch nicht oder nur unter enormen Schwierigkeiten in einer geschlossenen analytischen Form beschreibbar. Um allerdings alle Funktionswerte tabellarisch abzulegen wäre unbegrenzter Speicher und unbegrenzte Rechenzeit erforderlich. Beides steht nicht zur Verfügung, weswegen im weiteren Verlauf dieser Arbeit nur Approximationen an die echte GVCD behandelt werden können. Dazu wird nur eine diskrete Anzahl von Grauwerten betrachtet und Paare dieser Auswahl tabellarisch gespeichert. Damit ist die diskrete Schätzung der GVCD genaugenommen wieder auf die Berechnung der isotropen Gray Level Cooccurrence Matrix (IGLCM) reduziert. Der einzige Unterschied, der bleibt ist die freie Wahl der Quantisierung und die Betrachtung von allen für ein korrektes Abtasten der Daten erforderlichen Punkte-Paare mit Punktabstand r.

Anzahl möglicher Punktepaare Um die GVCD zu berechnen, müsste man alle möglichen Punktepaare des Signals berücksichtigen. Es gibt nun im Kontinuierlichen überabzählbar unendlich viele dieser Paare, so dass dies schlicht unmöglich ist. Die einzelnen Algorithmen, die im nächsten Kapitel vorgestellt werden, gehen mit dieser Problematik unterschiedlich um. Zwei Grundideen sind zu unterscheiden:

Auswahlverfahren Diese Verfahren betrachten nur eine Teilmenge aller möglichen Punktepaare. Implementierte Versionen sind das naive Verfahren, das alle Raumrichtungen mit einer definierten Schrittweite durchläuft und von jedem so betrachteten Punkt ausgehend nur endlich viele Nachbarpunkte mit Abstand $\left\|\vec{q}\right\|$ betrachtet. Die Schrittweite ist hier ein sehr wichtiger Parameter (siehe [Ron04]), der die Genauigkeit der Approximation und die Laufzeit sehr stark beeinflusst. Ein weiterer Algorithmus, der auf dem Auswahlverfahren basiert, ist die implementierte Monte Carlo Integration, bei der wiederholt zufällig Punktepaare bestimmt werden um anschließend den korrespondierenden Eintrag der GVCD um eins zu erhöhen. Zur Vergleichbarkeit mit den anderen Algorithmen ist klar, dass hier eine abschliessende Normierung auf die Anzahl Beispiele notwendig ist.

Kontinuierliche Verfahren Unter diesem Namen fasse ich alle Verfahren zusammen, die das diskrete Signal durch Anwendung der Fourier-Transformation implizit als kontinuierlich annehmen, und daher die Approximation auf Basis der Exponentialfunktionen des Frequenzspektrums vorgenommen wird. Dies führt zwangsläufig zu einem minimal anderen Ergebnis, ist aber gut mit den Ergebnissen der Auswahlverfahren vergleichbar. Unter diese Kategorie fallen der GVCD_CONV, GVCD_CORR, GVCD_FAST und der GVCD_SSI Algorithmus.

3.3. Algorithmen zur Berechnung der GVCD

Im folgenden werden die Algorithmen beschrieben, die zur Berechnung der Gray Value Cooccurrence Distribution entwickelt wurden.

3.3.1. Auswahlverfahren

Die in Abschnitt 3.2 erwähnten Auswahlverfahren sind sehr intuitiv und lassen sich ohne weitere Vorkenntnisse berechnen:

Der naive Algorithmus

Die wohl intuitivste Variante die Gray Value Cooccurrence Distribution zu berechnen, ist das Durchlaufen des gesamten (mehrdimensionalen) Signals mit einer bestimmten Schrittweite. An jeder Position werden dabei eine gewisse Anzahl Nachbarpunkte mit Abstand $\left\|\vec{q}\right\|$ betrachtet. Dabei darf keine Richtung bevorzugt werden. Im Fall eines 2-dimensionalen Signals ist das noch einfach zu erreichen, indem der Kreis mit Radius $\left\|\vec{q}\right\|$ in eine Anzahl gleicher Segmente unterteilt wird. Im 3-dimensionalen Fall ist dies leider nicht mehr so einfach möglich. Das Problem ist das Finden eines Gleichflächners mit *n* Flächen. Dies ist nur für wenige kleine *n* wirklich möglich. Das Problem wurde umgangen, indem anstelle dieser Suche eine Gauß-gefilterte Kugeloberfläche innerhalb eines festen Rasters berechnet wurde (wie bereits zur Berechnung der 2-Punkt-Invarianten in [RBS02] verwendet). Durch Wahl eines sehr schmalen Gauß-Filters, fallen die Werte mit wachsendem Abstand von der Kugeloberfläche schnell auf nahezu Null ab. Die verbleibenden Punkte sind reziprok exponentiell zum Abstand zur Kugeloberfläche gewichtet, wodurch zumindest eine richtungsunabhängige gleichmäßige Punkteverteilung erreicht wird. Zu beachten ist, dass die Gewichte der Punkte zur Beschreibung der Kurve/Oberfläche auf eine Summe von 1 normiert wurden.

Der naive Algorithmus ist in Alg. 1 in Pseudocode notiert:

Algorithm 1 GVCD_naiv

1: Initialize GVCD to 0
2: Precache a circle/sphere with radius $\left\|\vec{q}\right\|$
3: **for each** subpixel position $\vec{p}$ in the data block **X do**
4: **for each** point $\vec{s}$ on the circle/sphere surface with weight *w* **do**
5: Increase $\mathbf{gvcd}_{\left\|\vec{q}\right\|}\left(\mathbf{X}\left(\vec{p}\right), \mathbf{X}\left(\vec{p}+\vec{s}\right)\right)$ by *w*
6: **end for**
7: **end for**
8: Normalize GVCD

In Zeile 5 des Algorithmus wird das betrachtete Punktepaar in die GVCD eingetragen. Durch die Interpolation zwischen den diskreten Abtastpunkten des Signals erhält man reelle Zahlen, die so nicht als Indizes der GVCD verwendet werden können. Es gibt verschiedene Möglichkeiten dies zu lösen:

- Man erhöht den GVCD Eintrag, der am nächsten an dem berechneten Indexpaar liegt um *w*
- Man betrachtet alle direkt benachbarten Indexpaare der GVCD und verteilt *w* entsprechend dem Abstand vom „wirklichen" Indexpaar auf die vier anliegenden diskreten Indexpositionen.

Hier wurde die zweite Variante gewählt und implementiert. Sie erhöht den Berechnungsaufwand zwar um einen konstanten Faktor, führt aber zu genaueren Ergebnissen. Diese Methode wird als Fuzzy-Histogramm bezeichnet. Die Parallelen zu einem normalen Histogramm sind leicht zu sehen: Bei einer diskreten Anzahl Grauwerte, enthält ein Histogramm für jeden Grauwert gerade die Anzahl der Pixel eines Bildes, die den

entsprechenden Grauwert besitzen. Hat man nun keine diskret quantisierten Grauwerte, so kann man Grauwertintervalle bestimmen, die auf den gleichen Histogrammeintrag abgebildet werden (Variante 1) oder man kann den realen Grauwert linear auf die benachbarten Histogrammeinträge verteilen (Variante 2).

Die Laufzeit des naiven Algorithmus hängt stark von der Schrittweite entlang der Dimensionen des Signals ab, wie auch von der Anzahl betrachteter Punkte auf dem Kreis / der Kugeloberfläche. Sei d die Anzahl Dimensionen des Signals und N die Anzahl betrachteter Abtastpunkte entlang jeder Dimension. Sei weiterhin $r := \left\|\vec{q}\right\|$ der Abstand zwischen den Punkten und V die Grauwertauflösung der GVCD in eine Dimension, dann lässt sich die Laufzeit abschätzen durch:

$$\text{Laufzeit:} \quad T_{naiv}\left(N,d,r,V\right) = \underbrace{O\left(V^2\right)}_{\text{Init}} + \underbrace{O\left(N^d\right)}_{\text{Kreis}} + \underbrace{O\left(N^d\right)\cdot O\left(r^{d-1}\right)}_{\text{GVCD Berechnung}}$$

$$\text{Speicher:} \quad M_{naiv}\left(N,d,r,V\right) = O\left(N^d\right) + O\left(r^{d-1}\right) + O\left(V^2\right)$$

Da im Rahmen dieser Arbeit maximal dreidimensionale Daten verarbeitet werden, lässt sich dies mit $d = 3$ nach oben abschätzen. Die Initialisierung der GVCD und die Vorberechnung der Kugeloberfläche sind für die asymptotische Laufzeit vernachlässigbar.

Der Speicherplatzbedarf ist sehr moderat, da lediglich die GVCD, der Datensatz und ein Kreis / eine Kugeloberfläche gehalten werden müssen.

Monte Carlo Integrations Methode

Motivation: Die Berechnung der GVCD mit der naiven Methode ist sehr zeitaufwendig. Es gibt nun viele Anwendungsbereiche, in denen eine Randomisierung mit hinreichend vielen Einzelexperimenten zu Ergebnissen führt, die die deterministische Berechnung sehr gut approximieren aber nur einen Bruchteil des Rechenaufwands erfordern. Zur Berechnung von Haar-Integralen, einer Obermenge der hier verwendeten Grauwertinvarianten (vgl. Abschnitt 1.3), jedoch mit i. A. wesentlich höherem Berechnungsaufwand, wird die Verwendung von randomisierter Integration empfohlen und Experimente bestätigen das Potential dieser statistischen Methode (vgl. [SS99]).

Die Umsetzung in dieser Arbeit führte neben der Beschleunigung des Algorithmus auch zu einer wesentlich einfacheren Implementierung, wie sie in Alg. 2 gezeigt wird.

Algorithm 2 GVCD_MC

1: Initialize GVCD to 0
2: **for** $i = 1$ **to** #samples **do**
3: Randomly select a position $\vec{p}_1$ within the Data block $\mathbf{X}$
4: Randomly select an arc $\vec{\phi}$
5: $\vec{p}_2 = \vec{p}_1 + \mathbf{R}_{\vec{\phi}} \cdot \vec{q}$
6: Add 1 to $\mathbf{gvcd}_{\left\|\vec{q}\right\|}\left(\mathbf{X}\left(\vec{p_1}\right), \mathbf{X}\left(\vec{p}_2\right)\right)$
7: **end for**
8: Normalize GVCD

Der Algorithmus ist selbsterklärend, und im 2D Fall kann auch genau so vorgegangen werden, wie der Algorithmus es darstellt. Im 3D Fall ist die zufällige Wahl eines Raumwinkels etwas komplizierter. Daher wurde anstelle der Wahl des Winkels folgendes Verfahren verwendet um den zweiten Punkt zu ermitteln:

$x_1 := x_2 := x_3 := 1$
while $x_1^2 + x_2^2 + x_3^2 > 1$ **do**
  Select three equally distributed random numbers $x_1, x_2, x_3 \in \left[-1, 1\right]$
  $\vec{x} := \left(x_1, x_2, x_3\right)^T$
end while
$\vec{p}_2 := \left\|\vec{q}\right\| \cdot \frac{\vec{x}}{\|\vec{x}\|}$

Auch hier wurde beim Eintrag in die GVCD die Fuzzy-Histogramm Methode verwendet.

Das Verfahren ist einfach und robust. Die Laufzeit ist linear von der Anzahl Monte Carlo Experimente abhängig, die jedoch wiederum unter Berücksichtigung der Datengröße gewählt werden sollte, um einen hinreichend kleinen Approximationsfehler zu erreichen. Insofern ist eine Laufzeitabschätzung nicht ganz einfach. Eine genaue statistische Analyse wurde nicht durchgeführt, da der Algorithmus nur als „Wegweiser" dienen sollte, mit dem man schnell GVCDs schätzen kann, um die Ergebnisse anderer Verfahren grob verifizieren zu können.

In Abb. 3.1 sind die unterschiedlichen Vorgehensweisen des naiven Algorithmus im Vergleich mit der Monte Carlo Integration skizziert.

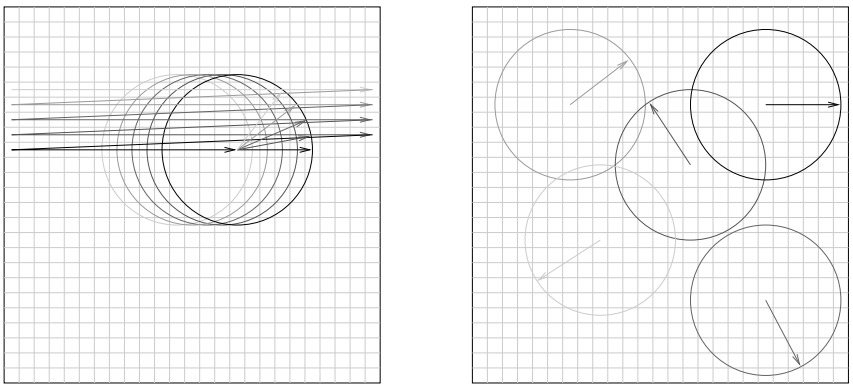


Abbildung 3.1.: Skizze zur Datenabtastung des naiven Algorithmus (links) und des Monte-Carlo Algorithmus (rechts)

Beide Verfahren haben den schwerwiegenden Nachteil, dass sie ein Sampling der Daten erfordern, was entweder bei wenigen Abtastpunkten zu sehr ungenauen Approximationen der kontinuierlichen GVCD führt oder wie bereits angesprochen bei feinerer Abtastung den Rechenaufwand untragbar erhöht.

3.3.2. Datenrepräsentation über Separation der Grauwerte

Um die kontinuierlichen Verfahren in geschlossenen Formeln beschreiben zu können, müssen zunächst Maskierungsfunktionen $m(x)$ definiert werden, mit denen einzelne Grauwerte des Datenblocks $\mathbf{X}$ maskiert

werden. Im kontinuierlichen Fall ist dies die Dirac-Funktion, wie in Formel 3.1 definiert.

$$\begin{aligned} \delta(x) &= 0 \quad \text{für } x \neq 0 \\ \int_{-\infty}^{\infty} \delta(x)\ \mathrm{d}x &= 1 \end{aligned} \tag{3.1}$$

Dies führt zu folgender geschlossener Form der Berechnung eines Eintrags der GVCD:

$$\mathbf{gvcd}_{||\vec{q}||}(v_1, v_2) = \int_G \delta\left(v_1 - \mathbf{X}\left(s_g\left(\vec{0}\right)\right)\right) \cdot \delta\left(v_2 - \mathbf{X}\left(s_g\left(\vec{q}\right)\right)\right)\ \mathrm{d}g \tag{3.2}$$

Dies entspricht gerade dem inneren Integral der Formel 1.8 in Abschnitt 1.3 über die Berechnung der Grauwertinvarianten.

Wie bereits in Abschnitt 3.2 diskutiert ist eine kontinuierliche Betrachtung aller Grauwerte impraktikabel. Das heißt, man muss sich auf eine (kleine) endliche Menge von Grauwerten beschränken um die GVCD in realistischer Zeit approximieren zu können. Die Dirac-Funktion eignet sich nun zur Maskierung nicht mehr, da die Pixel durch die Interpolation beliebige Werte zwischen minimalem und maximalem Grauwert des Datenblocks annehmen können. Würde man mit einem Dirac-Kamm Grauwerte selektieren, wären alle Grauwerte im Null-Bereich des Kamms verloren. Die Wahrscheinlichkeit, einen Grauwert genau zu treffen, geht gegen Null, so dass eine Maskierung mit dem Dirac-Impuls in diesem Fall eine Serie von nahezu leeren Masken liefern würde.
Um nun alternative Maskierungsfunktionen zu finden, die alle Grauwerte berücksichtigen, kann man die Herleitung des Dirac-Impulses zu Rate ziehen.

Der Dirac-Impuls ist der Grenzfall eines normierten Rechteckimpulses mit einer gegen Null gehenden Breite. Ändert man diese Breite, wird anstelle eines Grauwertes ein Grauwertbereich betrachtet. Allerdings wird der gesamte Grauwertbereich auf nur einen Wert abgebildet, was bei einer Rekonstruktion des Signals zu einer neuen Quantisierung führt. Ist das Originalsignal bereits quantisiert und wählt man die Rechteckimpulse so, dass gerade ein Grauwert zu einem Impuls korrespondiert, kann das Signal fehlerfrei rekonstruiert werden. Dies erfordert natürlich eine der Anzahl diskreter Grauwerte entsprechende Menge von Rechteckimpulsen. Leider war dies aus Gründen der Laufzeit nicht möglich, weswegen in diesem Schritt bei der Berechnung der GVCD ein kleiner Fehler in Kauf genommen werden musste. Die Definition des Rechteckimpulses ist in 3.3 notiert.

$$m^{\sigma}_{\text{rect}}(x) = \begin{cases} \dfrac{1}{\sigma} & \text{falls } x \in \left[-\dfrac{\sigma}{2}, \dfrac{\sigma}{2}\right) \\ 0 & \text{sonst} \end{cases} \tag{3.3}$$

Man sieht, dass für $\sigma \to 0$ der Rechteck-Impuls gegen den Dirac-Impuls geht.

Nun ist der Rechteckimpuls nur eine Möglichkeit den Dirac-Impuls herzuleiten. Eine für diesen Fall geschicktere Herleitung ist, im Gegensatz zur konstanten Rechteckfunktion, die „lineare" Dreiecksfunktion, wie sie in 3.4 definiert ist.

$$m^{\sigma}_{\text{triang}}(x) = \max\left\{\frac{1}{\sigma} \cdot \left(1 - \frac{|x|}{\sigma}\right), 0\right\} \tag{3.4}$$

Die Dreiecksfunktion enthält implizit die Fuzzy-Histogramm Methodik, die bereits vorgestellt wurde und ist damit mit den Auswahlverfahren am besten vergleichbar. Dies ist einer der Gründe, warum die Algorithmen diese Maskierungsfunktion als Standard verwenden. Auf andere Gründe wird mit Einführung der beiden verbleibenden getesteten Maskierungsfunktionen eingegangen.

Neben der Eigenschaft der Approximation der Dirac-Funktion, sollten die Maskierungsfunktionen nach Möglichkeit bandbegrenzt sein, um ein Aliasing der Masken auszuschließen. Dies war bei den bisher vorgestellten Maskierungsfunktionen nicht der Fall. Den Extremfall einer bandunbegrenzten Funktion bildet der Dirac-Impuls, dessen Frequenzspektrum konstant 1 ist.
Es gilt: Je „glatter" eine Funktion ist, desto weniger hohe Frequenzanteile enthält sie. Als Beispiel für diese Eigenschaft soll die Gauß-Kurve dienen. Sie ist beliebig oft stetig differenzierbar und ihr Frequenzspektrum ist selbst auch wieder eine Gauß-Kurve mit antiproportionaler Varianz. Um dies zu verdeutlichen, ist diese Korrespondenz in Abb. 3.2 dargestellt.

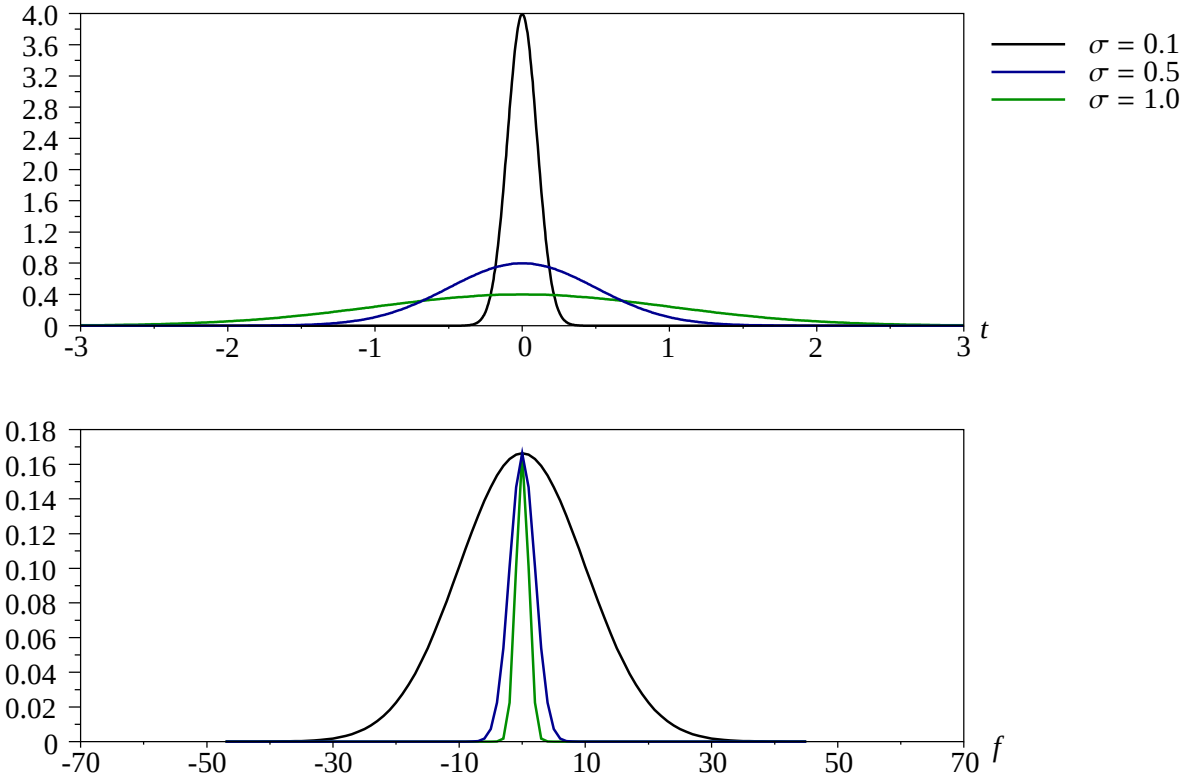


Abbildung 3.2.: Gauß-Kurven (oben) und zugehörige Frequenzspektren (unten)

In der Abbildung wird deutlich, dass Gauß-Kurven wieder auf Gauß-Kurven abgebildet werden. Das Maximum der Frequenzspektren liegt beim 0. Koeffizienten bei einem konstanten Wert, dem Mittelwert des betrachteten Ausschnitts der Gauß-Kurve im Ortsbereich. Steigt die Standardabweichung im Ortsbereich, sinkt sie im Frequenzbereich, wie erwartet. Diesen Sachverhalt kann man nun natürlich auch in Formeln fassen und erhält folgende Korrespondenz:

$$\mathcal{N}_\sigma(t) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\cdot\frac{t^2}{\sigma^2}} \quad \circ\!\!-\!\!\bullet \quad \overline{\mathcal{N}_\sigma(t)} \cdot e^{-\frac{1}{2}f^2\sigma^2} \tag{3.5}$$

Falls noch eine Mittelwertverschiebung hinzukommt, wird die Fourier-Transformierte komplex. Das Betragsspektrum bleibt aber in jedem Fall Gauß-förmig.

Nun lässt sich auch die Gauß-Kurve, wie in Formel 3.6 notiert, als Maskierungsfunktion verwenden. Im Grenzfall für $\sigma \to 0$ geht sie, ebenso wie die anderen Funktionen, in den Dirac-Impuls über, und sie ermöglicht die oben angesprochene Bandbegrenzung. Das Spektrum wird sogar schmaler, je höher die Stan-

dardabweichung der Gauß-Kurve gewählt wird. Das heißt, bei geringerer Grauwertauflösung werden hohe Frequenzen besser gefiltert, und Aliasing kann quasi nicht mehr auftreten.

$$m^{\sigma}_{\text{gauss}}(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\frac{x^2}{\sigma^2}} \tag{3.6}$$

Es gibt nun leider auch einen Nachteil der Maskierung mit der Gauß-Kurve: Die Summe einer Anzahl gegeneinander verschobener Gauß-Kurven ist, auch bei entsprechender Normierung, nicht in jedem Punkt 1, was für eine optimale Rekonstruktion des Originalsignals nach der Maskierung gelten muss. Es bleibt immer eine sinusförmige Schwankung um diesen Wert erhalten. Ein weiteres Problem sind die Randbereiche des Kamms. Da man nicht alle Grauwerte von $-\infty$ bis $+\infty$ abdecken kann, muss man sich auf ein Intervall, das das zu maskierende Signal enthält, beschränken. Da aber der Gauß nie auf Null abfällt, führt die Wahl eines Intervalls mit zu schmalem Rand zu einem erheblichen Approximationsfehler. Wählt man den Rand jedoch groß genug um den Fehler unter eine bestimmte Schranke sinken zu lassen, sind einige der betrachteten Gauß-Kurven nur noch wenig an der Repräsentation des Signals beteiligt, und die Schrittweite der Grauwerte wächst erheblich an, so dass insgesamt weniger Information aus dem Signal extrahiert wird. Diese Gründe ließen die Gauß-Maskierung auf realen gepaddeten Daten ausscheiden. Beide Probleme werden in Abb. 3.3 dargestellt.

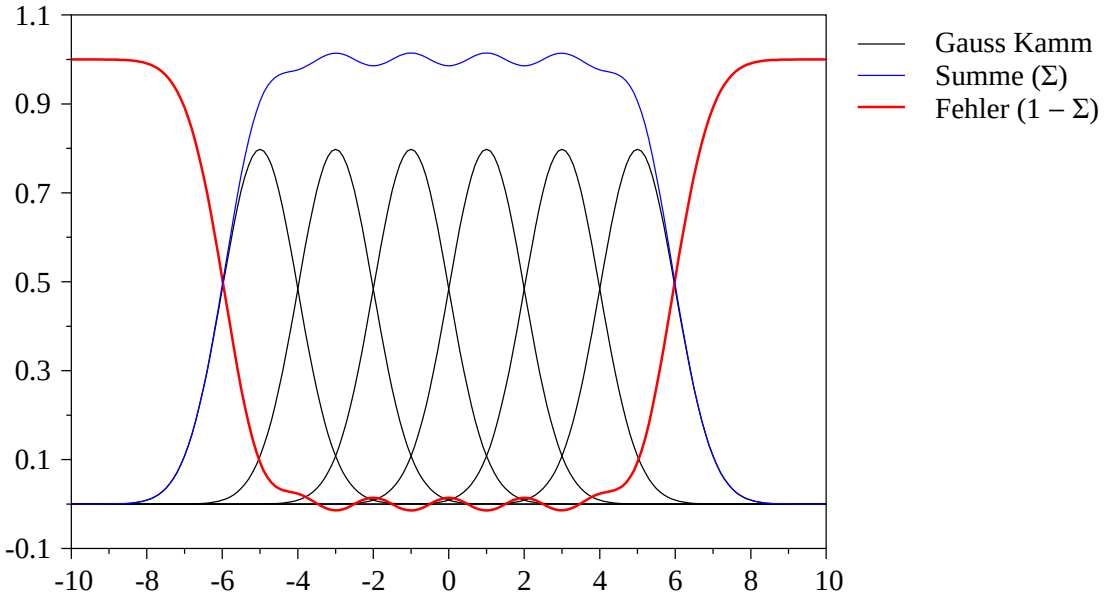


Abbildung 3.3.: Bleibender Fehler bei Gauß-Maskierung eines endlichen Grauwertbereichs

Die letzte implementierte Maskierungsfunktion ist etwas komplizierter, als die bisher vorgestellten. Die Motivation für die Entwicklung war die Kombination der einfachen Anwendbarkeit des in Gleichung 3.3 beschriebenen Rechteckimpulses mit der Bandbegrenzung der Gauß-Kurve. Dies wurde dadurch erreicht, dass die Unstetigkeiten des Rechtecks mit Hilfe der Sigmoid-Funktion stetig differenzierbar approximiert wurden, wie in Gleichung 3.7 notiert.

$$m^{\sigma,a}_{\text{sigmoid}}(x) = \frac{1}{\sigma} \cdot \left(\frac{1}{1 + e^{-a\cdot\left(x+\frac{\sigma}{2}\right)}} - \frac{1}{1 + e^{-a\cdot\left(x-\frac{\sigma}{2}\right)}} \right) \tag{3.7}$$

Zur Breite des Rechtecks σ kommt nun ein weiterer Parameter a hinzu, der die Flankensteilheit der neuen Maskierung beschreibt. Für $a \to \infty$ geht die Sigmoid-Maskierung in den Rechteckimpuls über. Je kleiner

a gewählt wird, desto weniger hohe Frequenzen enthält die Funktion. Der Nachteil dieser Maskierung liegt in der Einführung eines weiteren Parameters, der an die Breite σ und an die gewünschte Glattheit der Maskierungsfunktion angepasst werden muss. Je kleiner a gewählt wird, desto flacher wird der Sigmoid. Damit wird das Frequenzband, das verwendet wird zwar schmaler, aber die schon vom Gauß bekannten Probleme am Rand des betrachteten Intervalls treten wieder verstärkt auf, da die Kurve dann wieder sehr flach ausläuft.

Für die Berechnungen der GVCD-Approximationen in dieser Arbeit wurden alle angesprochenen Maskierungen ausprobiert und nach dem Vergleich der Ergebnisse der Berechnung der 2-Punkt-Invarianten entschieden, die Dreiecksmaskierung zu verwenden, da diese auch bei skalierten Daten zu vergleichbaren Ergebnissen führte. Grundsätzlich eignen sich aber alle vorgestellten Funktionen zum Zweck der Berechnung von Kernfunktionen, mit ihren jeweiligen Vor- und Nachteilen.

Die digitale Signalverarbeitung bietet noch ein Reihe möglicher Standard-Fensterfunktion, die gute Eigenschaften bezüglich der Bandbegrenzung haben und trotzdem im Zeitbereich begrenzt sind. Beispiele hierfür sind das Hann-, das Hamming- oder das Kaiser-Fenster. Normiert man diese Fenster auf ein Integral von 1, sind sie genauso für die Maskierung geeignet. Weitere Informationen zu diesen und anderen Fenster-Funktionen sind z. B. [PB87] zu entnehmen.

Da alle vorgestellten Funktionen nun nicht kontinuierlich für jeden Grauwert verwendet werden sollen, sondern Grauwertbereiche maskieren, ist eine abschließende Normierung durch Multiplikation mit der Breite dieses Bereiches sinnvoll. Will man aus den Masken das Originalsignal rekonstruieren, ist so nur noch eine gewichtete Summe der Masken mit ihren korrespondierenden mittleren Grauwerten erforderlich. Die entsprechend normierten Maskierungsfunktionen sind in den Gleichungen 3.8-3.11 notiert.

$$\hat{m}^{\sigma}_{\text{rect}}(x) = \begin{cases} 1 & \text{falls } x \in \left[-\frac{\sigma}{2}, \frac{\sigma}{2}\right) \\ 0 & \text{sonst} \end{cases} \tag{3.8}$$

$$\hat{m}^{\sigma}_{\text{triang}}(x) = \max\left\{1 - \frac{|x|}{\sigma}, 0\right\} \tag{3.9}$$

$$\hat{m}^{\sigma}_{\text{gauss}}(x) = \sqrt{\frac{2}{\pi}} \cdot e^{-\frac{1}{2}\frac{x^2}{\sigma^2}} \tag{3.10}$$

$$\hat{m}^{\sigma,a}_{\text{sigmoid}}(x) = \frac{1}{1 + e^{-a\cdot\left(x+\frac{\sigma}{2}\right)}} - \frac{1}{1 + e^{-a\cdot\left(x-\frac{\sigma}{2}\right)}} \tag{3.11}$$

Um die Wirkung der einzelnen Maskierungen auf die Daten zu verdeutlichen, wurde exemplarisch in Abb. 3.4 ein realer Datensatz maskiert. Es wurden jeweils 8 Masken erzeugt, die verschiedene Grauwerte hervorheben.

Analog zum kontinuierlichen Fall (vgl. Gleichung 3.2) berechnet sich die GVCD im Diskreten unter Zuhilfenahme einer der obigen Maskierungsfunktionen folgendermaßen:

$$\mathbf{gvcd}_{\|\vec{q}\|}(v_1, v_2) = \int_G \hat{m}\left(v_1 - \mathbf{X}\left(s_g\left(\vec{0}\right)\right)\right) \cdot \hat{m}\left(v_2 - \mathbf{X}\left(s_g\left(\vec{q}\right)\right)\right) \, \mathrm{d}g \tag{3.12}$$

Dies ist eine Umformulierung des naiven Algorithmus, unter Zuhilfenahme der obigen Masken. Allerdings wäre der Algorithmus, der sich direkt aus obiger Formel ergibt, noch wesentlich ineffizienter als die vorgestellte Variante, weswegen auf eine Implementierung verzichtet wurde. Um dies zu verdeutlichen möchte ich trotzdem kurz den Pseudocode vorstellen und eine kleine Laufzeit- und Speicherbedarfsanalyse anstellen.

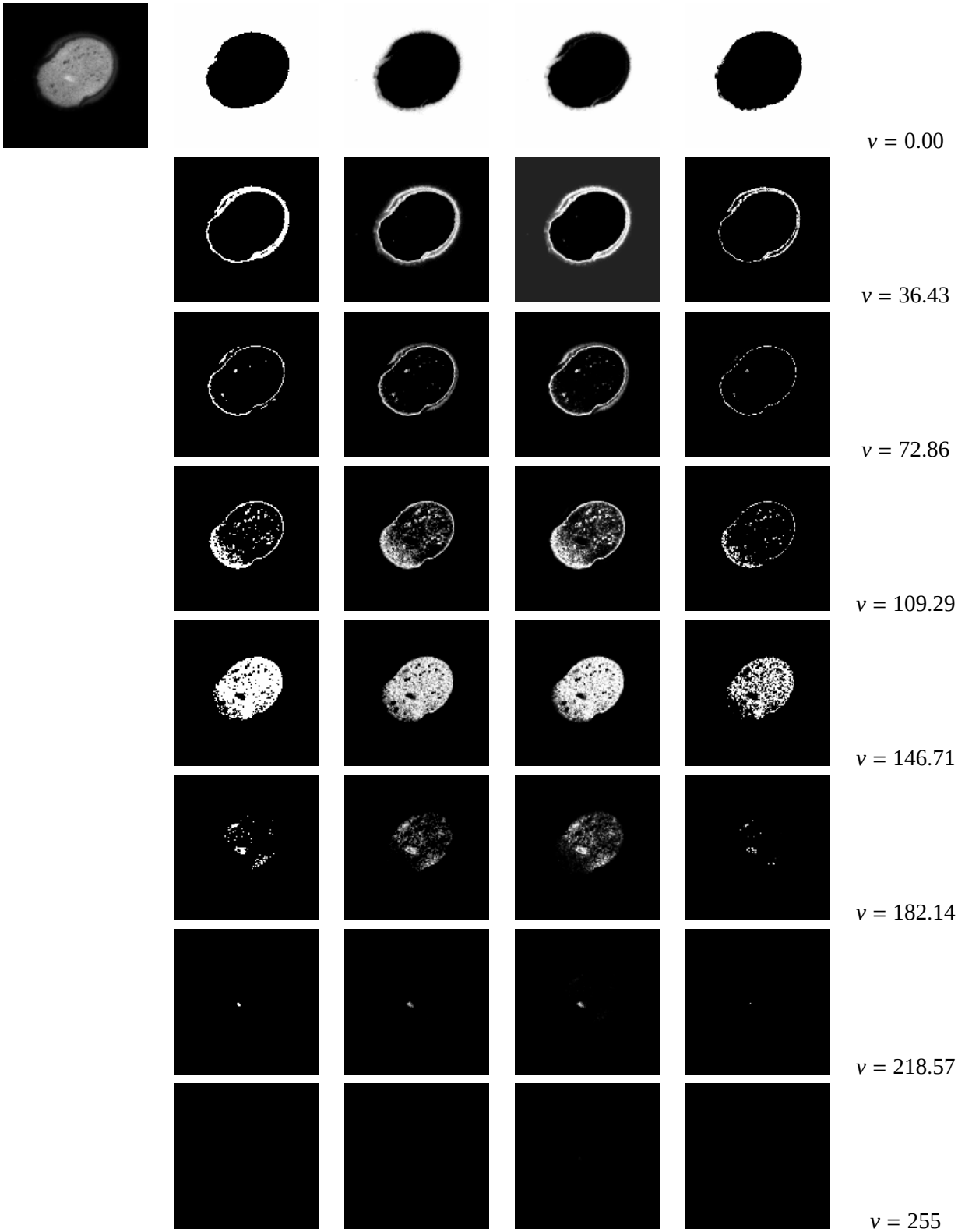


Abbildung 3.4.: Anwendung der Maskierungsfunktionen auf einen realen Datensatz

Algorithm 3 GVCD_NAIV2

```
1: Initialize GVCD to 0
2: Precache a circle/sphere surface with radius ||q⃗||
3: for each gray value v do
4:    Create gray value mask M_v(x⃗) := m̂(v − X(x⃗))
5: end for
6: for each gray value v_1 do
7:    for each gray value v_2 do
8:       for each subpixel position p⃗ in the data block do
9:          for each point s⃗ on the circle/sphere surface with weight w do
10:            gvcd_||q⃗||(v_1, v_2) ← gvcd_||q⃗||(v_1, v_2) + M_{v_1}(p⃗) · M_{v_2}(p⃗ + s⃗)
11:         end for
12:      end for
13:   end for
14: end for
15: Normalize GVCD
```

Seien die Parameter definiert, wie zuvor, dann sind die Kosten des obigen Algorithmus bezüglich Zeit und Speicherbedarf:

$$\text{Laufzeit:} \quad T_{naiv_2}(N,d,r,V) = \underbrace{O\left(V^2\right)}_{\text{Init}} + \underbrace{O\left(N^d\right)}_{\text{Kreis}} + \underbrace{V \cdot O\left(N^d\right)}_{\text{Maskierung}} + \underbrace{V^2 \cdot O\left(N^d\right) \cdot O\left(r^{d-1}\right)}_{\text{GVCD Berechnung}}$$

$$\text{Speicher:} \quad M_{naiv_2}(N,d,r,V) = O\left(V^2\right) + V \cdot O\left(N^d\right) + O\left(r^{d-1}\right)$$

Um die Nachteile der Auswahlverfahren auf den Daten zu vermeiden, wurde auf Basis der Maskierung eine Reihe weiterer Algorithmen entwickelt.

Zunächst möchte ich auf eine direkt einsichtige Lösung eingehen. Die Idee besteht darin, den Kreis (bzw. die Kugeloberfläche) wie zuvor vorzuberechnen, ihn aber dann, anstelle der bisher verwendeten speichersparenden Datenstruktur, in einer Matrix der Dimension der Eingangsdaten zur Verfügung zu stellen . Im weiteren werden Matrizen, die in dieser Art erzeugte Kreise oder Kugeloberflächen enthalten, mit $\mathbf{C}_r$ bezeichnet.
Um die Notation zu vereinfachen, wird folgende Kurzschreibweise für die maskierten Signale eingeführt:

$$\mathbf{M}_v(\vec{x}) := \hat{m}\left(v - \mathbf{X}(\vec{x})\right) \tag{3.13}$$

Mit den vorberechneten Kreisen kann nun mit Hilfe folgender Formel ein schnellerer Algorithmus entwickelt werden.

$$\mathbf{gvcd}_r(v_1,v_2) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \mathbf{M}_{v_1}(\vec{x}) \cdot \left(\mathbf{M}_{v_2} * \mathbf{C}_{||\vec{q}||}\right)(\vec{x})\ \mathrm{d}x_1 \ldots\ \mathrm{d}x_d \tag{3.14}$$

Betrachtet man obige Berechnungsvorschrift, erkennt man eine große Ähnlichkeit mit der Berechnung der 2-Punkt-Grauwert-Invarianten aus Abschnitt 1.3 Formel 1.5. In der Tat besteht der einzige Unterschied darin, dass die Daten zur Berechnung der GVCD zuvor maskiert wurden. Diese Beobachtung hat für die weiteren Schritte zur schnellen Berechnung der GVCD eine große Bedeutung. Wenn nämlich jeder Eintrag der GVCD nichts anderes als eine 2-Punkt-Invariante darstellt, können alle Algorithmen zur schnellen Berechnung dieser Invarianten auch auf die Berechnung der GVCD angewendet werden. Dieser Gedankengang führt uns zu den kontinuierlichen Berechnungsverfahren der GVCD.

3.3.3. Kontinuierliche Verfahren

Die erste Implementation auf dieser Basis ist die direkte Umsetzung von Formel 3.14 in Algorithmus 4

Algorithm 4 GVCD_CONV

1: Calculate a circle/sphere $\mathbf{C}_{||\vec{q}||}$
2: **for each** gray value v **do**
3: Calculate gray value mask $\mathbf{M}_v\left(\vec{x}\right) := \hat{m}\left(v - \mathbf{X}\left(\vec{x}\right)\right)$
4: **end for**
5: **for each** gray value v_1 **do**
6: $\mathbf{M}_{v_1,||\vec{q}||} := \mathcal{F}^{-1}\left\{\mathcal{F}\left\{\mathbf{M}_{v_1}\right\} \cdot \mathcal{F}\left\{\mathbf{C}_{||\vec{q}||}\right\}\right\}$
7: **for each** gray value v_2 **do**
8: $\mathbf{gvcd}_{||\vec{q}||}\left(v_1, v_2\right) := \sum \mathbf{M}_{v_1,||\vec{q}||} \cdot \mathbf{M}_{v_2}$
9: **end for**
10: **end for**
11: Normalize GVCD

Für jedes Grauwertpaar wird in diesem Algorithmus eine 2-Punkt-Grauwert-Invariante berechnet, was zu folgenden Kosten des Algorithmus führt:

$$\begin{aligned}
\text{Laufzeit:} \quad T_{CONV}\left(N,d,r,V\right) &= \underbrace{O\left(N^d\right)}_{\text{Kreis}} + \underbrace{V \cdot O\left(N^d\right)}_{\text{Maskierung}} + \underbrace{V \cdot O\left(N^d \log N\right) + V^2 \cdot O\left(N^d\right)}_{\text{GVCD Berechnung}} \\
&= V \cdot O\left(N^d \log N\right) + V^2 \cdot O\left(N^d\right) \\
\text{Speicher:} \quad M_{CONV}\left(N,d,r,V\right) &= O\left(V^2\right) + O\left(N^d\right) + V \cdot O\left(N^d\right)
\end{aligned}$$

Wie in Abschnitt 1.3 Formel 1.6 bereits erwähnt, gibt es alternativ zu obiger Formel auch die Möglichkeit über die Autokorrelation des Signals die 2-Punkt-Grauwert-Invarianten zu berechnen. Analog dazu, kann man mit Hilfe der Maskierungsfunktionen über Verwendung der Kreuzkorrelation zu den GVCD Einträgen gelangen. Dies ist in Formel 3.15 dargestellt.

$$\mathbf{gvcd}_{||\vec{q}||}\left(v_1, v_2\right) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \mathbf{C}_{||\vec{q}||}\left(\vec{x}\right) \cdot \left(\mathbf{M}_{v_1} \otimes \mathbf{M}_{v_2}\right)\left(\vec{x}\right) \, \mathrm{d}x_1 \ldots \, \mathrm{d}x_d \tag{3.15}$$

Zur Veranschaulichung betrachtet man am besten die Dirac-Maskierung und zur weiteren Vereinfachung ein eindimensionales Signal. Die Kreuzkorrelierte wird genau an den Stellen Werte ungleich Null aufweisen, zu denen in den Masken Impulse mit entsprechendem Abstand existieren. Dies wird in Abb. 3.6 deutlich, in der der gesamte Prozess an einem sehr einfachen eindimensionalen Signal nachvollzogen wird.

Wie die Faltung in Formel 3.14, lässt sich die Berechnung der Kreuzkorrelierten schnell im Frequenzbereich durchführen:

$$f\left(t\right) \otimes g\left(t\right) = \mathcal{F}^{-1}\left\{\mathcal{F}\left\{f\left(t\right)\right\} \cdot \mathcal{F}\left\{g\left(t\right)\right\}^*\right\} \tag{3.16}$$

Der sich aus dieser Idee ergebende Algorithmus ist in Alg. 5 dargestellt.

Betrachtet man den Algorithmus genauer, erkennt man, dass die Fourier-Transformation im Gegensatz zu Alg. 4 nun in der tiefsten Schleifenebene berechnet wird. Dies lässt vermuten, dass der Algorithmus erheblich langsamer ist, als sein Gegenstück, das die gleiche Berechnung mit Hilfe der Faltung durchführt.

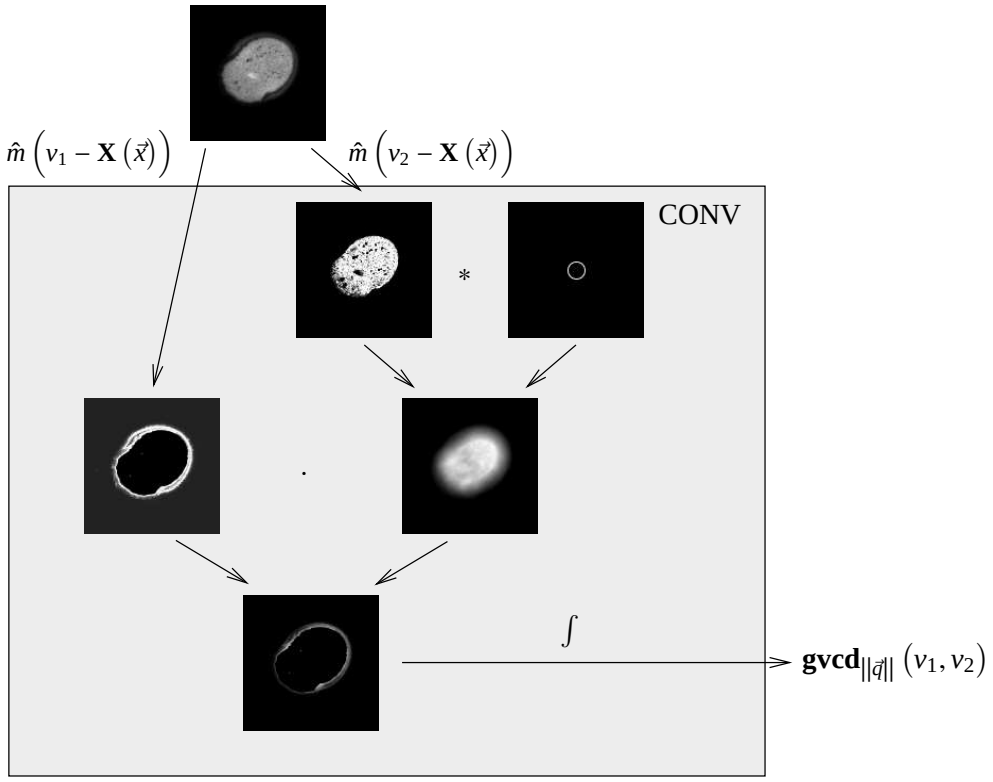


Abbildung 3.5.: Schematischer Ablauf des gvcd_CONV Algorithmus am Beispiel eines Ahornpollen

Algorithm 5 GVCD_CORR

Prechache a circle/sphere $\mathbf{C}_{||\vec{q}||}$
for each gray value v **do**
  Calculate gray value mask $\mathbf{M}_v\left(\vec{x}\right) := \hat{m}\left(v - \mathbf{X}\left(\vec{x}\right)\right)$
  $\widetilde{\mathbf{M}}_v := \mathcal{F}\left\{\mathbf{M}_v\right\}$
end for
for each gray value v_1 **do**
  for each gray value v_2 **do**
    $\mathbf{M}_{v_{1,2}} := \mathcal{F}^{-1}\left\{\widetilde{\mathbf{M}}_{v_1} \cdot \widetilde{\mathbf{M}}^*_{v_2}\right\}$
    $\mathbf{gvcd}_{||\vec{q}||}\left(v_1, v_2\right) := \sum \mathbf{C}_{||\vec{q}||} \cdot \mathbf{M}_{v_{1,2}}$
  end for
end for
Normalize GVCD

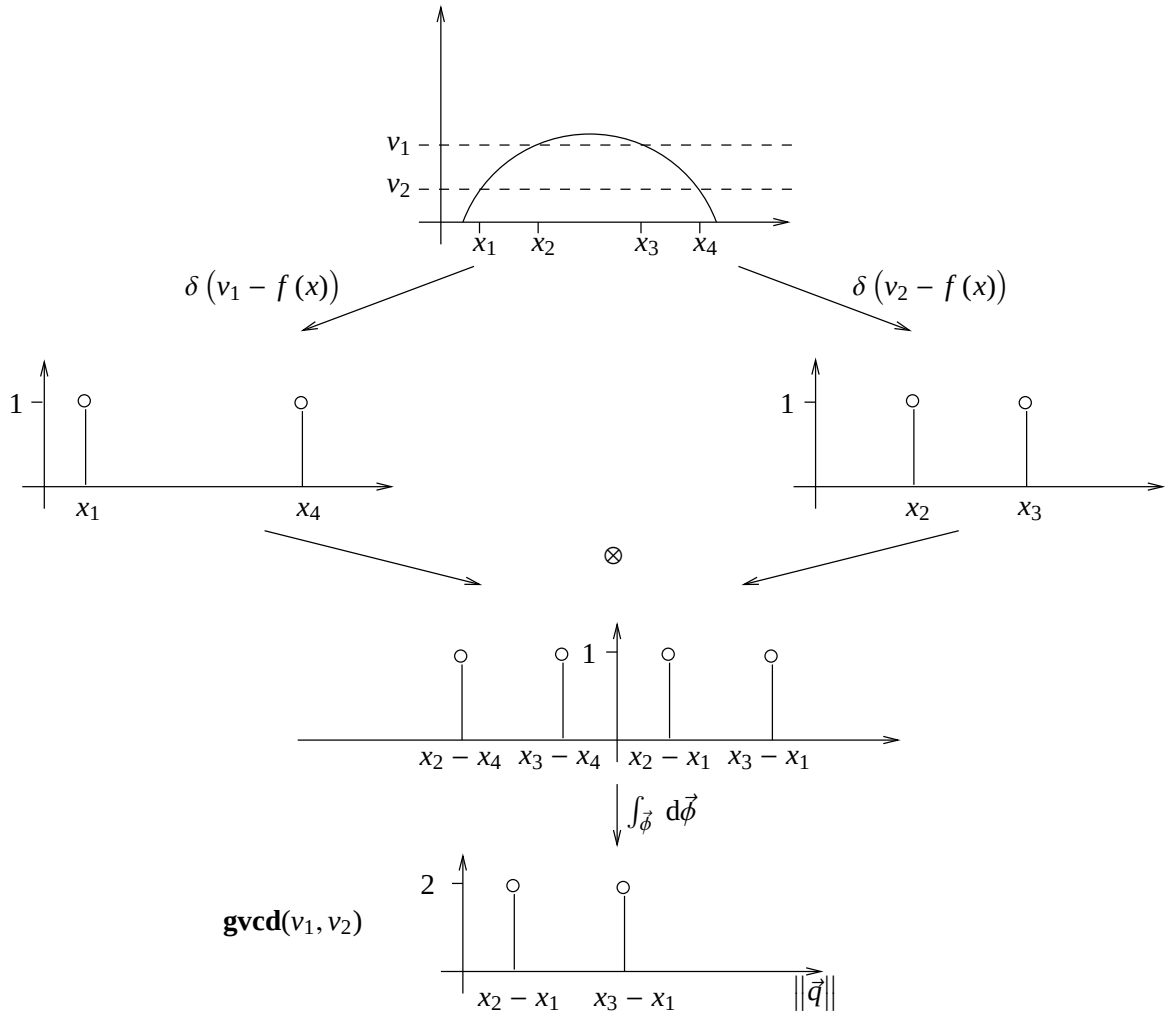


Abbildung 3.6.: Vom Signal zur GVCD

Die theoretischen asymptotischen Kosten des GVCD_CORR-Algorithmus liegen bei:

$$\text{Laufzeit:} \quad T_{CORR}\left(N,d,r,V\right) = \underbrace{O\left(N^d\right)}_{\text{Kreis}} + \underbrace{V \cdot O\left(N^d \log N\right)}_{\text{Maskierung}} + \underbrace{V^2 \cdot O\left(N^d \log N\right)}_{\text{GVCD Berechnung}}$$

$$\text{Speicher:} \quad M_{CORR}\left(N,d,r,V\right) = O\left(V^2\right) + O\left(N^d\right) + V \cdot O\left(N^d\right)$$

Trotz dieser Laufzeitverschlechterung, liefert der Ansatz über Korrelation die Basis zu einem schnellen Algorithmus.

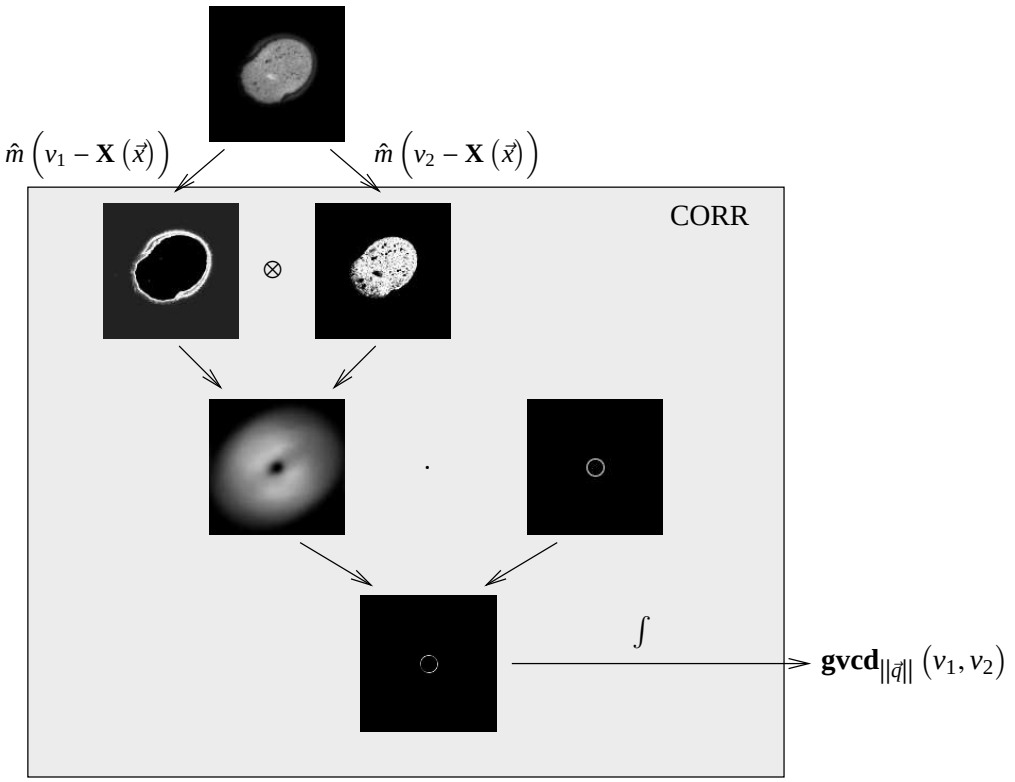


Abbildung 3.7.: Schematischer Ablauf des gvcd_CORR Algorithmus am Beispiel eines Ahornpollen

Der teure Rechenschritt ist die Berechnung der inversen Fourier-Transformation in jedem Schleifendurchlauf. Bei genauem Hinsehen und ein wenig Fourier-Theorie erkennt man jedoch, dass man die gesamte Berechnung im Frequenzbereich durchführen kann und die inverse Fourier Transformation damit einspart.

Anstatt die Kreuzkorrelierte im Ortsbereich mit dem Kreis (der Kugel) zu multiplizieren und anschließend über das ganze Signal zu integrieren, kann auch direkt im Frequenzbereich mit der Fourier-Transformierten des Kreises (der Kugel) multipliziert werden mit anschließender Mittlung über alle Frequenzen.
Das entspricht einer Faltung der Kreuzkorrelierten mit dem Kreis im Ortsbereich mit anschließender Selektion des Koordinatenursprungs.
Begründung: Nach der Faltung enthält jeder Punkt des resultierenden Signals den Wert des Integrals der umliegenden Punkte mit Abstand $\left\|\vec{q}\right\|$. Der Wert im Ursprung ist für uns von Interesse, denn er ist so gerade das Integral über die Werte entlang des gesuchten Kreises (bzw. auf der Kugeloberfläche). Zieht man die Fourier-Theorie zu Rate, erkennt man, dass der nullte Fourier-Koeffizient eines Signals gerade den Mittelwert desselben beschreibt. Mit Hilfe der Dualität zwischen Zeit- und Frequenzbereich, lässt sich auch

die Umkehrung dieser Aussage zeigen und man erkennt, dass der gesuchte Wert im Ursprung des Signals ebenso durch den Mittelwert des Spektrums beschrieben wird. Eine Integration über das mit dem Kreisspektrum multiplizierte Spektrum der Kreuzkorrelierten mit anschließender Normierung liefert also den gesuchten Eintrag für die GVCD, ohne die inverse Fourier-Transformation durchführen zu müssen. Da im Frequenzbereich dabei nur punktweise Multiplikationen auszuführen sind, gilt das Kommutativgesetz, was den folgenden Algorithmus 6 durch geschickte Umsortierung noch ein wenig beschleunigt.

Algorithm 6 GVCD_FAST

1: Prechache a circle/sphere $\mathbf{C}_{\|\vec{q}\|}$
2: $\widetilde{\mathbf{C}}_{\|\vec{q}\|} := \mathcal{F}\left\{\mathbf{C}_{\|\vec{q}\|}\right\}$
3: **for each** gray value v **do**
4: Calculate gray value mask $\mathbf{M}_v\left(\vec{x}\right) := \hat{m}\left(v - \mathbf{X}\left(\vec{x}\right)\right)$
5: $\widetilde{\mathbf{M}}_v := \mathcal{F}\left\{\mathbf{M}_v\right\}$
6: **end for**
7: **for each** gray value v_1 **do**
8: $\widetilde{\mathbf{M}}_{v_1,\|\vec{q}\|} := \widetilde{\mathbf{M}}_{v_1} \cdot \widetilde{\mathbf{C}}_{\|\vec{q}\|}$
9: **for each** gray value v_2 **do**
10: $\widetilde{\mathbf{M}}_{v_{1,2},\|\vec{q}\|} := \widetilde{\mathbf{M}}_{v_1,\|\vec{q}\|} \cdot \widetilde{\mathbf{M}}^*_{v_2}$
11: $\mathbf{gvcd}_{\|\vec{q}\|}\left(v_1, v_2\right) := \text{mean}\left(\widetilde{\mathbf{M}}_{v_{1,2},\|\vec{q}\|}\right)$
12: **end for**
13: **end for**
14: Normalize GVCD

Die Kosten dieses neuen Algorithmus liegen dann bei:

$$\begin{aligned} \text{Laufzeit:} \quad T_{FAST}\left(N,d,r,V\right) &= \underbrace{O\left(N^d \log N\right)}_{\text{Kreis}} + \underbrace{V \cdot O\left(N^d \log N\right)}_{\text{Maskierung}} + \underbrace{V^2 \cdot O\left(N^d\right)}_{\text{GVCD Berechnung}} \\ \text{Speicher:} \quad M_{FAST}\left(N,d,r,V\right) &= O\left(V^2\right) + O\left(N^d\right) + V \cdot O\left(N^d\right) \end{aligned}$$

Dies war der aktuellste Algorithmus, der vollständig implementiert wurde. Verfolgt man die Entwicklung der 2-Punkt-Invarianten-Berechnung weiter, gibt es jedoch einen weiteren Algorithmus, der noch erheblich schneller ist. Dieser Algorithmus wurde von Qing Wang entwickelt und wird unter dem Namen „Expression of Gray Scale Invariants with Monomial Kernels in Fourier Space" veröffentlicht werden (siehe [Wan05]). Die Theorie hinter dem Algorithmus ist etwas trickreich, basiert aber auf der Korrelationsmethode zur Berechnung der 2-Punkt-Invarianten.

Beobachtung: Das Frequenzspektrum des Kreises (der Kugeloberfläche) ist rotationssymmetrisch, das heißt, entlang eines Kreises haben die Fourier-Koeffizienten den gleichen Wert.

Es bietet sich nun an, die Werte entlang der Kreislinien gleicher Radien aufzuintegrieren und damit das zugrundeliegende mehrdimensionale Signal auf eine Dimension zu reduzieren. Dieses eindimensionale Spektrum kann mit dem eindimensionalen Gewichtungsvektor des korrespondierenden Kreisspektrums multipliziert werden, und man erhält durch Integration über alle Koeffizienten mit korrekter Normierung wieder den selben Wert.
Zu beachten ist in diesem Fall, dass die Projektion eines Kreisspektrums auf eine Dimension gerade die erste Bessel-Funktion nullter Ordnung liefert. Im Fall von dreidimensionalen Daten, entspricht die Projektion

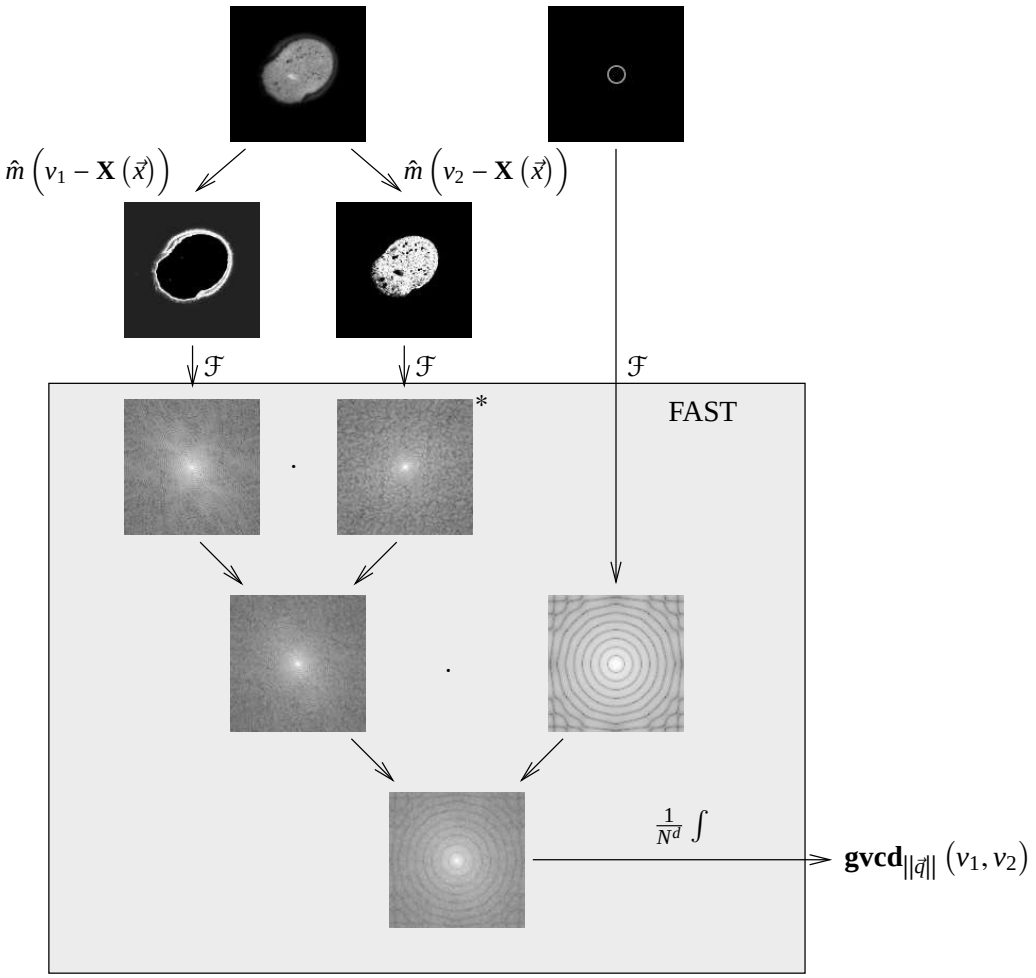


Abbildung 3.8.: Schematischer Ablauf des gvcd_FAST Algorithmus am Beispiel eines Ahornpollen

der Fourier-Transformierten einer Kugel gerade einer $\frac{\sin x}{x}$-Funktion. Diese kann mit wenig Aufwand und sehr geringem Speicherbedarf vorberechnet werden. Ein weiterer großer Vorteil dieses Verfahrens ist, dass die Kreislinie nicht mehr approximiert werden muss, sondern dass der Kreis direkt im Frequenzspektrum bereitgestellt wird und damit die Lösung exakter ist und die benachbarten Radien abgesehen von datenbedingten Korrelationen, dekorreliert sind. Natürlich ist auch in diesem Fall für die Projektion der Spektren der Kreuzkorrelierten eine Approximation notwendig, und im Randbereich der Spektren treten neue unerwünschte Nebeneffekte auf, die jedoch durch die geringen Beträge der in diesen Bereichen angesiedelten Fourier-Koeffizienten zu vernachlässigen sind.

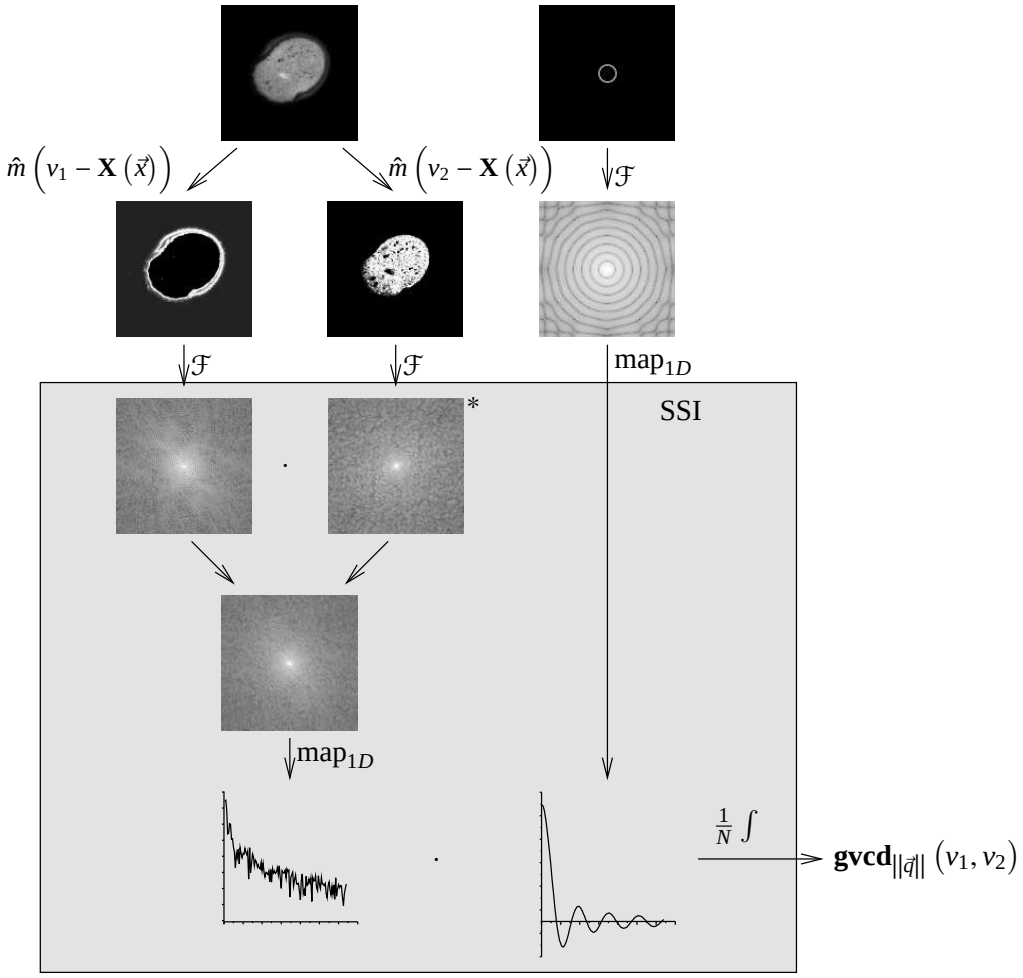


Abbildung 3.9.: Schematischer Ablauf des gvcd_SSI Algorithmus am Beispiel eines Ahornpollen

Der große Vorteil der frühen Berechnung der Kreuzkorrelierten zwischen den Masken ist die Möglichkeit, mit nur sehr geringem Aufwand GVCDs für verschiedene Punktabstände $\left\|\vec{q}\right\|$ zu berechnen, da im letzten Schritt der GVCD-Berechnung über die Kugeloberfläche mit Radius $\left\|\vec{q}\right\|$ integriert wird. Dies kommt insbesondere dem SSI-Algorithmus zu Gute, da hier durch Vorberechnung verschiedener Bessel-, bzw. $\frac{\sin x}{x}$-Funktionen mit einer einmal auf eine Dimension abgebildeten Kreuzkorrelierten entsprechende GVCD-Einträge in $O(N)$ berechnet werden können. Im Vergleich dazu benötigt der gvcd_FAST Algorithmus dafür Zeit in der Größenordnung von $O(N^d)$.

Wie bereits gesagt, wurde dieser Algorithmus hier nicht implementiert, sondern eine leicht abgewandelte Form modular integriert. Der Unterschied zwischen Originalalgorithmus und modifizierter Variante liegt ausschließlich in der Normierung, die aus dem SSI-Algorithmus ausgelagert wurde und so in die Verantwortung der GVCD-Berechnung überging. Dies war notwendig, da sonst anstelle einer Normierung auf das Originalsignal, auf die maskierten Signale normiert und damit das Ergebnis verfälscht würde. Leider sind die Ergebnisse nicht identisch mit denen der anderen Berechnungsverfahren, weswegen für Vergleiche und auch für die Experimente dieser Algorithmus ausgeklammert und besonders behandelt wurde.

3.4. Skalierung der Daten: Eine weitere GVCD-Dimension

Neben der Wahl verschiedener Abstände $\left\|\vec{q}\right\|$ zur Berechnung von 2-Punkt-Invarianten hat sich die Verwendung eines Multiskalenansatzes als vielversprechend erwiesen. Die Begründung hierfür findet sich auch im Vortrag über die 2-Punkt-Invarianten und der Diskussion über deren Vollständigkeit (siehe [Ron04]). Dort wird gezeigt, dass eine Vorfilterung der Signale Mehrdeutigkeiten eliminieren kann.

Alle bestehenden Algorithmen zur Berechnung von 2-Punkt-Invarianten implementieren eine Vorfilterung mit Hilfe eines Gauß-Filters mit anschließender Skalierung der Daten. In den Implementationen ist der Parameter σ für diese Skalierung verantwortlich. Wird $\sigma = 0$ gewählt, werden die Originaldaten verarbeitet, wie bis jetzt im Rahmen dieser Arbeit diskutiert. Für $\sigma \in \mathbb{N}^+$, wird das Signal mit einem Gauß mit Standardabweichung σ gefiltert und anschließend um Faktor 2σ verkleinert.

Der Grund für diese Wahl ist im Spektrum des Signals zu suchen. Würde das Signal nicht mit dem Gauß vorgefiltert, würde ein Aliasing auftreten, das durch die Bandbegrenzung durch Anwendung des Gauß verhindert wird. Bei obiger Wahl der Standardabweichung werden die Beträge der im skalierten Bild höchsten Frequenzen auf ca. 8% des Originalbetrags abgesenkt. Der verbleibende Fehler ist für geringe Skalierungen zu vernachlässigen, da dieser Frequenzbereich schon im Vorfeld nur geringe Beträge aufweist. Je breiter jedoch der Gauß wird, desto schmaler wird sein Spektrum, so dass der Fehler bei hohen Skalierungen zunimmt. Dies kann kompensiert werden, indem das Pad vor der Skalierung entsprechend erweitert wird, so dass die Gauß-Filterung ohne störende Randeffekte durchgeführt werden kann. Dies wurde wegen der ohnehin schon sehr langsamen Berechnung aber nicht berücksichtigt, muss aber bemerkt werden, da dadurch die Ergebnisse gerade im Bereich hoher Skalierungen nicht 100%ig korrekt sind. Da dies jedoch bereits in den gegebenen Algorithmen nicht berücksichtigt wurde, sind die Ergebnisse wieder direkt vergleichbar, was ein weiterer Grund war, diesen Fehler zu vernachlässigen.

Alle bisherigen Implementationen hatten nun nur die Möglichkeit der Wahl von ganzzahligen Werten für σ. Diese Einschränkung wurde zur Berechnung der GVCD aufgehoben, da sie theoretisch unbegründet ist. Über den Sinn der Wahl Nicht-Ganzzahliger Skalierungen lässt sich jedoch streiten, da schon die Betrachtung aller sinnvollen ganzzahligen Skalierungen aus Gründen der Rechenzeit impraktikabel ist.

Jeder der im Abschnitt 3.3 vorgestellten Algorithmen, ist in der Lage einen Parametersatz bestehend aus Skalierungen $\sigma_1, \ldots, \sigma_k$ und korrespondierenden Punktabständen $\left\|\vec{q}\right\|_1, \ldots, \left\|\vec{q}\right\|_k$ zu verarbeiten und die entsprechenden GVCD's zu berechnen. Die Parameter werden dazu in einer eigenen netCDF Datei zur Verfügung gestellt.

3.5. Berechnung von Mehr-Kanal GVCDs

Bisher wurde die Berechnung der GVCD auf einem Datensatz mit nur einem Kanal vorgestellt. Die Berechnung läßt sich jedoch einfach auf Mehrkanal-Aufnahmen eines Objekts erweitern. Zum Beispiel lassen sich RGB-Farbwerte auf diese Weise analysieren, indem zunächst GVCDs auf jedem Kanal einzeln, unabhängig von den anderen, berechnet werden, dann jedoch auch für jede Kanalkombination. Dadurch erhält man ein wesentlich größeres Spektrum an aussagekräftigen Merkmalen, die durch ausschließlich getrennte Betrachtung der Kanäle nicht erreichbar gewesen wäre.

Die kombinierte GVCD aus zwei Kanälen zu berechnen, wenn bereits Algorithmen für einzelne Kanäle existieren, ist sehr einfach und intuitiv. Betrachten wir hierzu noch einmal als Beispiel die Berechnungsvorschrift für einen GVCD Eintrag nach der Korrelationsmethode (Gl. 3.15):

$$\mathbf{gvcd}_r\left(v_1, v_2\right) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \mathbf{C}_{\|\vec{q}\|}\left(\vec{x}\right)\left(\mathbf{M}_{v_1} \otimes \mathbf{M}_{v_2}\right)\left(\vec{x}\right)\ \mathrm{d}x_1 \ldots\ \mathrm{d}x_d$$

Verwendet man nun anstelle eines Signals $\mathbf{X}$, zwei Signale $\mathbf{X}_i$ und $\mathbf{X}_j$ mit Grauwertmasken $\mathbf{M}_v^i$, bzw. $\mathbf{M}_v^j$ und substituiert die Einträge in der Formel, erhält man:

$$\mathbf{gvcd}_r^{i,j}\left(v_1, v_2\right) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \mathbf{C}_{\|\vec{q}\|}\left(\vec{x}\right)\left(\mathbf{M}_{v_1}^i \otimes \mathbf{M}_{v_2}^j\right)\left(\vec{x}\right)\ \mathrm{d}x_1 \ldots\ \mathrm{d}x_d \tag{3.17}$$

Diese Formel ist eine Verallgemeinerung der Ein-Kanal-GVCD und für $i = j$ erhält man wieder die alte Vorschrift, so dass man mit Algorithmen zur Berechnung von 2-Kanal-GVCDs die einfachen GVCDs weiterhin ohne Änderungen berechnen kann.

Bemerkung: Bei Mehrkanal-GVCDs geht die Symmetrie der GVCD verloren, weswegen nun alle Einträge wirklich berechnet werden müssen. Aus diesem Grund ist es trotzdem sinnvoll zwei verschiedene Methoden zur Berechnung von 1- bzw. 2-Kanal GVCDs zu haben, da bei der 1-Kanal-GVCD nur $\frac{V\cdot(V+1)}{2}$ Werte berechnet werden müssen im Gegensatz zu allen V^2 bei der 2-Kanal-GVCD. In der Größenordnung ändert dies nichts, aber ein Faktor 2 in der Berechnungsgeschwindigkeit ist bei der recht langsamen Berechnung nicht bedeutungslos.

Bei k Kanälen erhält man so $\frac{k\cdot(k+1)}{2}$ Kanalkombinationen, die modulo Symmetrie zur ersten Hauptdiagonalen unterschiedliche GVCDs ergeben.

Beispiel 3.5.1:
Kanäle: A, B, C

Kombinationen: AA, AB, AC, BB, BC, CC

3.6. Praktische Laufzeiten der Algorithmen

Für Laufzeitvergleiche auf realer Hardware wurden künstlich erzeugte Datensätze verwendet. Diese enthalten nur gleichverteiltes, mittelwertfreies Rauschen, im Zahlenbereich $(-1, +1)$ um datenabhängige Laufzeitschwankungen durch Verwendung normalisierter und denormalisierter Fließkommazahlen zu vermeiden. Die Laufzeiten beschränken sich dabei auf die reine Berechnung der GVCD. Die Vorberechnung der Masken und eventuell benötigter Kreise/Kugeln wurde nicht berücksichtigt. Der Testrechner war ein Pentium III Coppermine 650MHz mit 256kB 2nd Level Cache und 320MB Arbeitsspeicher. Die Algorithmen sind in C++ implementiert unter Verwendung von blitz++ Arrays als Datenstruktur und der fftw (Fastest Fourier Transform of the West) Bibliothek in der Version 2.1.3 zur Berechnung der Fourier-Transformationen.

Die Diagramme in Abb. 3.10 beschreiben die Laufzeit der Algorithmen in Abhängigkeit des gewählten Abstandes $\left\|\vec{q}\right\|$, der Anzahl Grauwerte V und der Anzahl Abtastpunkte N. Für die Monte-Carlo-Integration wurde ein Test pro Datenelement als Samplingparameter verwendet.

3.6.1. Schlussfolgerungen

Aus der realen Laufzeitanalyse, wird klar, dass sich für verschiedene Anforderungen an die GVCD unterschiedliche Algorithmen eignen. Die wichtigen Parameter für die Wahl des Algorithmus sind zum einen die verwendeten Punktabstände $\left\|\vec{q}\right\|$ und zum anderen die Grauwertauflösung, also die Anzahl der betrachteten Grauwerte V. Alle Algorithmen sind linear von der Datengrösse abhängig, weswegen dieser Parameter auf die Algorithmen-Wahl keinen Einfluss hat. Bei der Entscheidung, werden hier auch der Monte-Carlo- und der SSI-Algorithmus nicht berücksichtigt, da im Fall der Monte-Carlo-Integration nur eine approximative Lösung erreicht wird, und im Fall der Scale Space Invariants (SSI), keine mit den restlichen Verfahren vergleichbaren Ergebnisse zu erzielen waren.

Der naive Algorithmus ist für sehr kleine Punktabstände aber sehr große Grauwertauflösungen gut geeignet, allerdings auch nur für diesen Sonderfall. Für alle anderen Fälle eignet sich die Berechnung über Korrelation im Frequenzbereich am Besten. Man sieht aber, dass mit höherdimensionalen Daten der Laufzeitunterschied zwischen der Berechnung über schnelle Faltung und über schnelle Korrelation im FAST Algorithmus abnimmt. Es wäre interessant diesen Trend bei höherdimensionalen Daten weiterzuverfolgen, wenn das Verfahren zum Beispiel auf zeitliche Sequenzen von volumetrischen Daten angewendet werden soll.

Eine Kombination aus hohen Grauwertauflösungen und großen Punktabständen ist mit den entwickelten Algorithmen nicht effizient berechenbar. Dazu wäre es erforderlich die quadratische Abhängigkeit von der Anzahl der Grauwerte aus den kontinuierlichen Verfahren zu eliminieren, dies ist aber nur möglich, wenn auf die Maskierung verzichtet wird. Wenn also Algorithmen existieren, die für beide Dimensionen der Laufzeitproblematik die GVCD schnell berechnen können, ist dies eine neue Klasse von Algorithmen, die im Laufe dieser Arbeit nicht gefunden wurde.

Die Verwendung des beschriebenen Multiskalenansatzes entschärft die Problematik, da durch verschiedene Skalierungsstufen implizit verschiedene Punktabstände verwendet werden. Beschränkt man sich also auf echte Punktabstände von 1 - 2 Pixel und skaliert dafür die Daten, kann mit dem naiven Verfahren recht effizient eine große Anzahl von GVCDs berechnet werden. Dies ist ein erstaunliches Ergebnis, da nun der intuitivste Algorithmus durch seine Unabhängigkeit von der Grauwertauflösung in kleinen, aber wichtigen Parameterbereichen auch der schnellste ist.

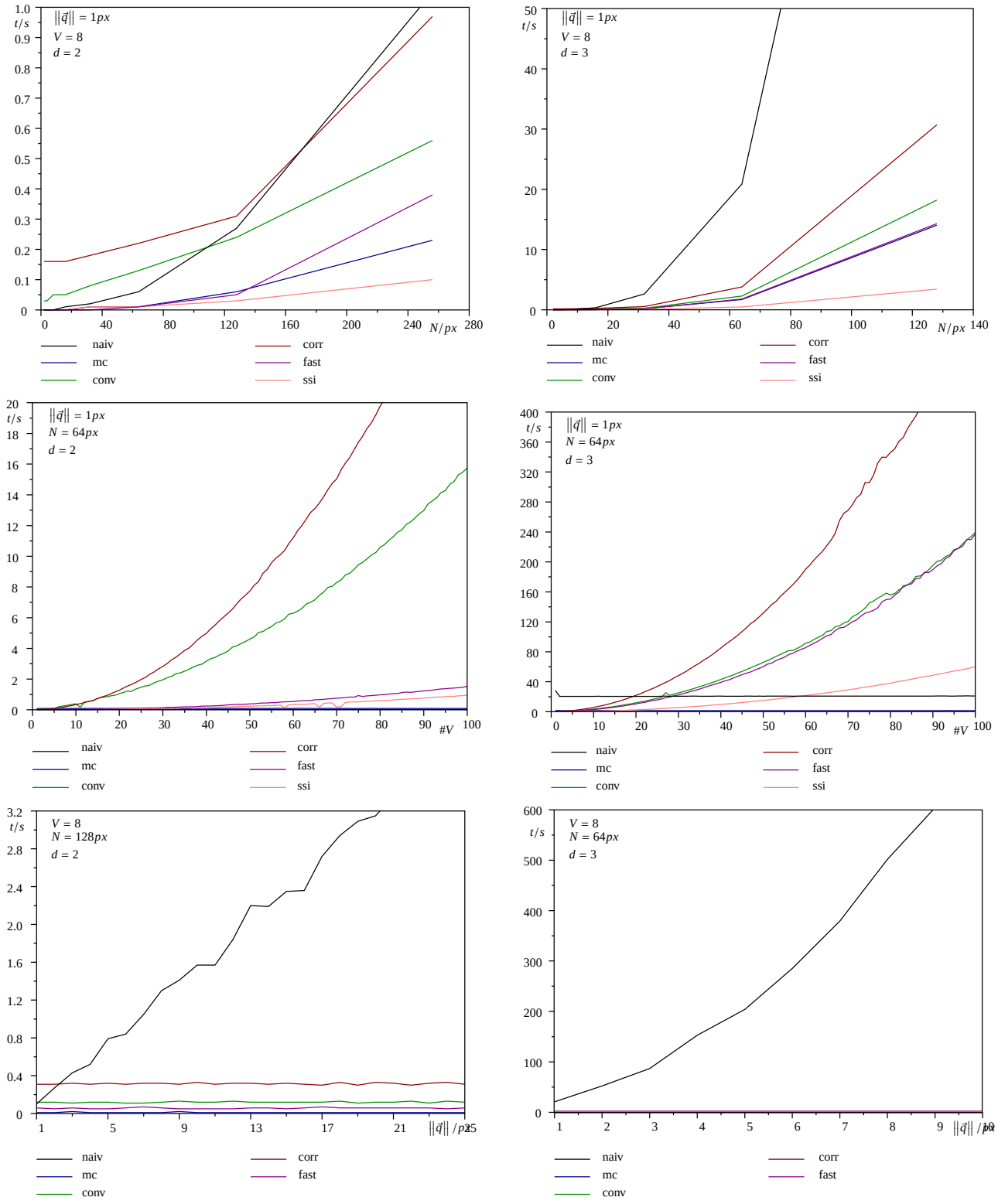


Abbildung 3.10.: Laufzeiten der Algorithmen zur Berechnung der GVCD in Abhängigkeit der Datengröße N, der Anzahl Grauwerte V und der Punktabstände $\|\vec{q}\|$

3.7. Beispiele für GVCDs

Nach der Einführung der theoretischen Grundlagen der GVCDs und der Möglichkeiten der Berechnung, möchte ich einige Beispiele für GVCDs, der in Kapitel 2 beschriebenen Datensätze, aufzeigen.

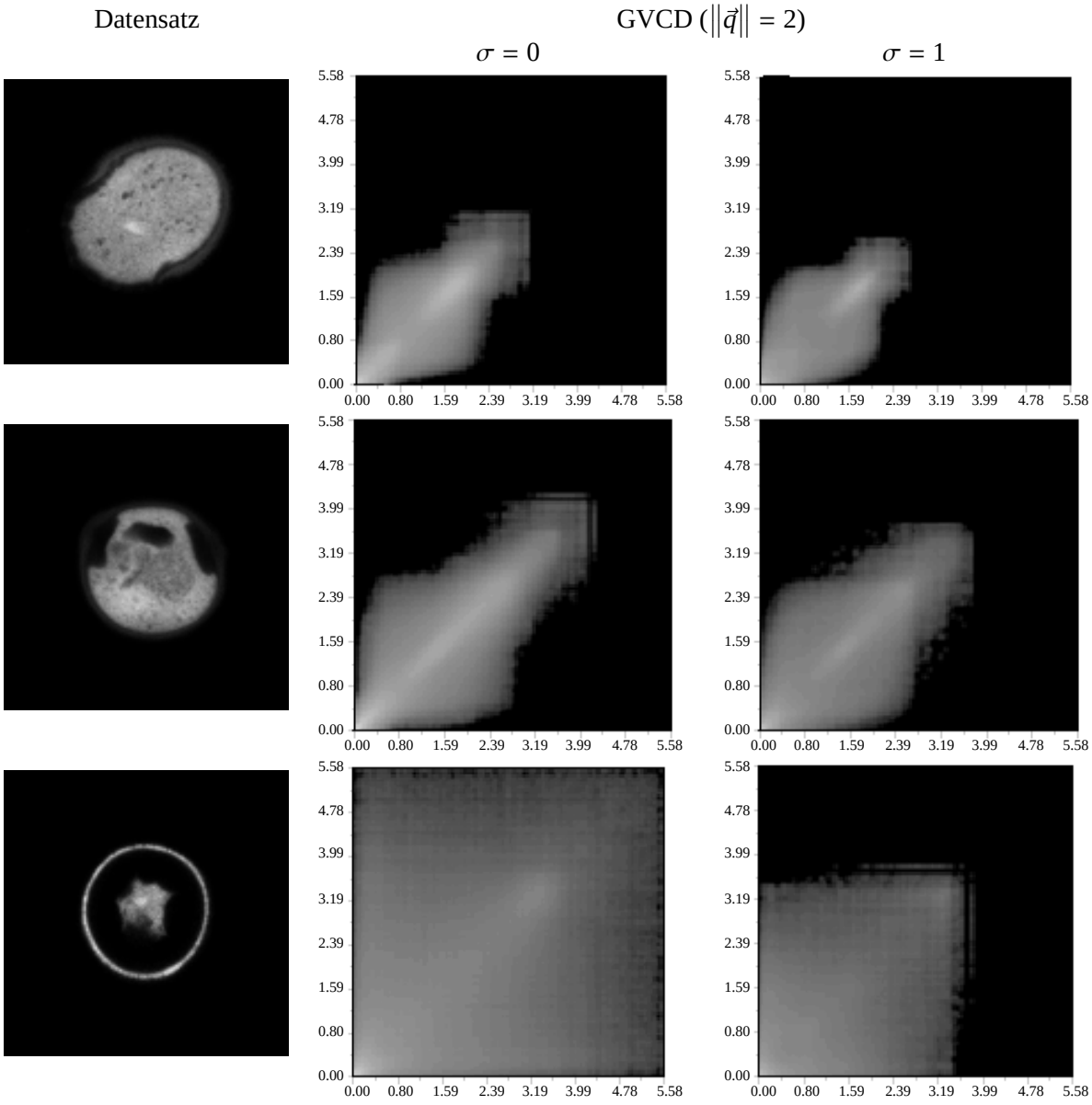


Abbildung 3.11.: Beispiele für GVCDs auf skalierten dreidimensionalen Testdaten

In Abbildung 3.11 sind die mit dem naiven Algorithmus berechneten GVCDs mit 128 Graustufen und einem Punktabstand $\|\vec{q}\|$ von 2 Pixel zu sehen. In der linken Spalte sind die Originaldatensätze abgebildet. Diese wurden auf ihre jeweilige Standardabweichung normiert um Kontrastschwankungen durch Ausbleichen der Pollen zu kompensieren. Der in den GVCDs abgebildete Grauwertbereich ist für alle Abbildungen gleich gewählt um, einen direkten Vergleich zu ermöglichen. Die mittlere Spalte zeigt die GVCDs der Eingabedaten in Originalgröße, die rechte die, der um Faktor 2 verkleinerten Daten.

Auffällig ist die GVCD des Eibenpollen in der letzten Zeile. Die GVCD ist beinahe homogen über alle Grauwertpaare. Die große Ausdehnung entlang der Diagonalen ergibt sich aus der Normierung, da durch die geringe Grauwertsumme des Eibenpollen mit einem hohen Faktor normiert wurde. Die enorme Breite hingegen ist erstaunlich, da sie zeigt, dass nahezu alle Grauwertkombinationen mit einem Abstand von nur 2 Pixeln im Datensatz auftreten, was bei den anderen Datensätzen nicht zu beobachten ist. Eine Erklärung hierfür sind die sehr feinen Strukturen, die eine direkte Nachbarschaft sehr unterschiedlicher Grauwerte fördern.

Um die GVCDs in dieser Form abbilden zu können, wurde eine logarithmische Skala verwendet, da wie bereits diskutiert der Dynamikumfang der GVCDs enorm groß ist.
Abbildung 3.12 zeigt die mit dem naiven Algorithmus berechneten GVCDs mit 128 Grauwerten und einem Punktabstand von 2 Pixel, wie bereits bei den zuvor gezeigten 3D Beispielen. Auch hier wurden die Daten zunächst auf ihre Standardabweichung normiert und dann die GVCDs aus allen Kanalkombinationen berechnet. Die Maxima der GVCDs liegen jeweils beim Grauwert 0, der auch zum Padden verwendet wurde. Betrachtet man die GVCDs, die den Durchlichtkanal berücksichtigen, sieht man dass hier ein Großteil der Grauwerte des Originalsignals im negativen Bereich liegen. Dies liegt an der Vorverarbeitung, in der der Hintergrund abgezogen wurde, um ein Padding überhaupt erst zu ermöglichen.

Da es sich um Durchlichtaufnahmen handelt, ist klar, daß ein Teil des Lichts an schwer lichtdurchlässigen Stellen absorbiert wird und so abgebildete Strukturen dunkler sind als der Hintergrund. Bei den Pollen ist dieser Effekt zwar auch dominant, erklärt aber nicht den gesamten Abbildungsprozess. Die unterschiedlich dichten Medien innerhalb der Pollen führen zu komplizierten Lichtbrechungen, die ebenso lokal abschatten wie auch aufhellen können. Dies ist auch der Hauptgrund, warum in den gängigen Verfahren auf die Verwendung des Durchlichtkanals für die automatische Erkennung verzichtet wurde, da diese Nebeneffekte zu Transformationen führen, die weit über die euklidische Bewegung und kleinere Deformationen hinausgehen.

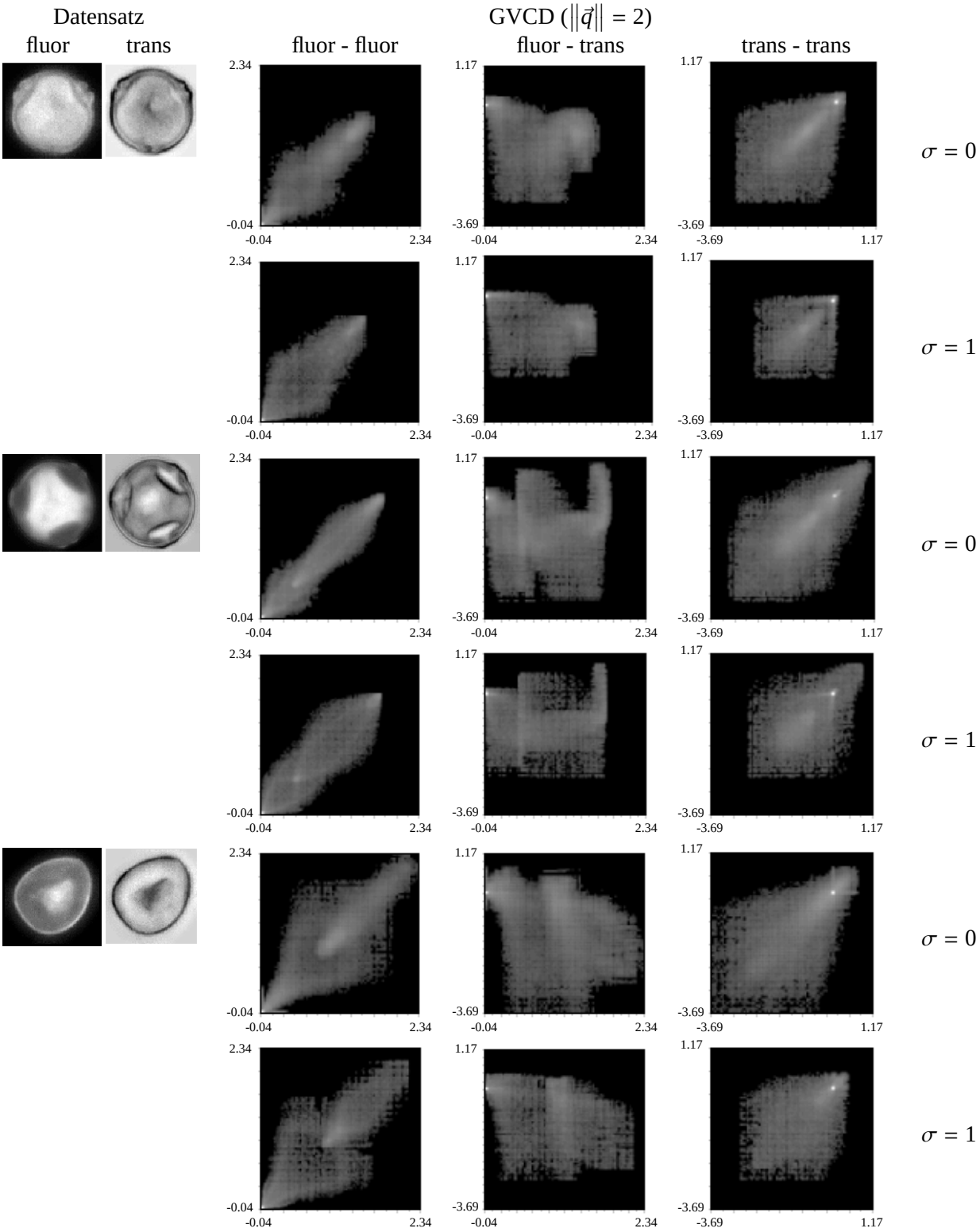


Abbildung 3.12.: Beispiele für 2-Kanal GVCDs auf skalierten zweidimensionalen Testdaten

4. Bestimmung von Kernfunktionen

4.1. Vergleich zwischen direkter Berechnung und indirekter Berechnung über die GVCD

Nach der Berechnung der GVCD nach einem der Verfahren aus Kapitel 3 besteht der zweite Schritt zur Berechnung von 2-Punkt-Invarianten in einer Gewichtung der Einträge der GVCD mit der Kernfunktion $f\left(v_1, v_2\right)$. Die einfachste Kernfunktion, mithilfe derer alle obigen Algorithmen verifiziert wurden, ist die Funktion $f\left(v_1, v_2\right) = v_1 \cdot v_2$. Unter Verwendung dieser Kernfunktion entspricht die aus der GVCD berechnete Invariante der Standard 2-Punkt-Invariante, wie sie in Abschnitt 1.3 eingeführt wurde, abgesehen von den diskutierten Quantisierungsfehlern.

Um dies zu untermauern und außerdem auf spezielle Eigenschaften der einzelnen Algorithmen einzugehen, folgt ein direkter Vergleich der 2-Punkt-Invarianten berechnet über die direkte Berechnungsvorschrift aus Abschnitt 1.3 mit den gleichen Invarianten unter Verwendung der gewichteten GVCDs. In diesem Fall ist bei der Berechnung der GVCD darauf zu achten, dass die Normierung gleich gewählt ist, wie bei der direkten Berechnung der 2-Punkt-Invarianten. Um dies zu erreichen wurden die Daten in jeder Skalierungsstufe auf eine Grauwertsumme von 1 normiert.

Testdaten Als Referenz wurden die Testdatensätze aus Kapitel 2 verwendet, bestehend aus drei dreidimensionalen (128^3 Voxel) und drei zweidimensionalen Pollendatensätzen (512×576 Pixel) (s. Abb. 2.1). Die Datensatzgrößen sind dabei inklusive Pad, das in der Abbildung nicht gezeigt wird

Parametersatz $P = \left\{ \left(0,2\right), \left(1,2\right), \left(2,2\right), \left(4,2\right), \left(8,2\right), \left(16,2\right) \left(32,2\right) \right\}$
Jeder Eintrag des Parametersatzes P hat dabei die Form $\left(\sigma, \left\|\vec{q}\right\|\right)$

In Abb. 4.1 ist der Vergleich zwischen direkter Berechnung und indirekter Berechnung mit Hilfe der GVCD zu sehen. Trotz einer sehr geringen Grauwertauflösung von nur 8 Grauwerten ist der Unterschied im Plot nicht erkennbar. Erst im Differenzplot zwischen den Invarianten aus beiden Verfahren sieht man den Unterschied.

Man erkennt zunächst, dass die direkt über die GVCD berechneten Invarianten sich im schlimmsten Fall in der 4. Nachkommastelle unterscheiden. Im Allgemeinen werden sie erheblich besser approximiert. Die ausgeprägte Spitze im Differenzplot des 3D-Falls bei $\sigma = 8$ (dem 5. Feature) ergibt sich aus der realen Verkleinerung der Datensätze, im Gegensatz zur Berechnung der GSI Invarianten, bei der die Datengröße konstant gehalten wird und sich damit das Padding verbreitert. Dieser Fehler könnte durch gleiche Behandlung bei Berechnung der GVCD noch vermieden werden, wurde jedoch aus Gründen der Rechenzeit in Kauf genommen. Im 2D Fall tritt dieser Fehler nicht auf, da dort die Daten von vorneherein sehr großzügig gepaddet waren und so auch bei realer Verkleinerung der Daten kein Aliasing auftritt. Der wachsende Fehler bei starken Verkleinerungen ist darauf zurückzuführen, daß die GVCD kaum noch Werte ungleich Null enthielt und dadurch nur noch wenige, für den verbleibenden Grauwertbereich verhältnismäßig grob abgetastete Grauwerte in die Berechnung mit einflossen.

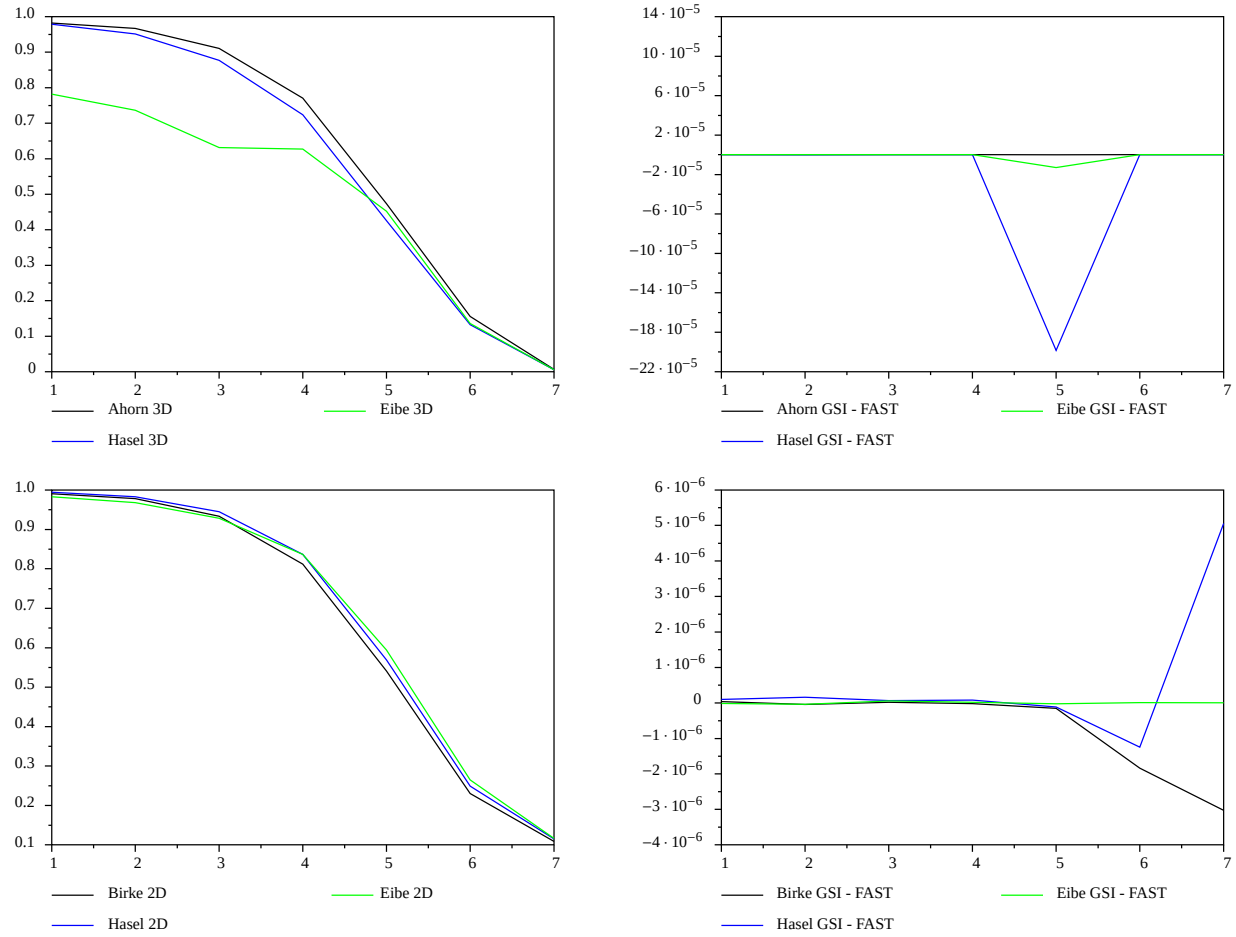


Abbildung 4.1.: Vergleich zwischen direkter Berechnung der 2-Punkt-Invarianten und Berechnung mit Hilfe der GVCD

Insgesamt ist jedoch zu bemerken, dass trotz der sehr groben Grauwertauflösung von nur 8 Grauwerten, eine unerwartet gute Approximation an die direkt berechneten 2-Punkt-Invarianten zu erreichen war, die immerhin ein Spektrum von 256 Graustufen betrachten.

4.2. Transformationskorrespondenzen

Um zu verstehen, wie nun anhand der berechneten GVCDs möglichst optimale Kernfunktionen bestimmt werden können, deren Wirkung auch ohne Berechnung der GVCD direkt erreichbar ist, möchte ich einige Betrachtungen anstellen, welche Transformationen auf den Eingabedaten, welche Änderungen in der GVCD nach sich ziehen.

4.2.1. Grauwerttransformationen

Da die GVCD als eine Verbundverteilung der Grauwerte der Eingabedaten aufgefasst werden kann, ist die einfachste Form von Transformationen die Transformation der Grauwerte der Originaldaten. Die Funktion $f : \mathbb{R} \to \mathbb{R}$ ist dabei eine beliebige Funktion, die auf die Grauwerte angewendet wird. Mit $\vec{f} : \mathbb{R}^n \to \mathbb{R}^n$ wird diese Funktion durch elementweise Anwendung auf Vektoren erweitert.

Signal	GVCD
$\mathbf{X}$	$\left(\mathbf{gvcd}_{\|\vec{q}\|}\left(\mathbf{X}\right)\right)\left(v_1, v_2\right) = \int_G \delta\left(v_1 - \mathbf{X}\left(s_g\left(\vec{0}\right)\right)\right) \cdot \delta\left(v_2 - \mathbf{X}\left(s_g\left(\vec{q}\right)\right)\right) \mathrm{d}g$
$\mathbf{X} + \alpha \cdot \mathbf{1}$	$\left(\mathbf{gvcd}_{\|\vec{q}\|}\left(\mathbf{X} + \alpha \cdot \mathbf{1}\right)\right)\left(v_1, v_2\right) = \left(\mathbf{gvcd}_{\|\vec{q}\|}\left(\mathbf{X}\right)\right)\left(v_1 - \alpha, v_2 - \alpha\right)$
$\alpha \cdot \mathbf{X}$	$\left(\mathbf{gvcd}_{\|\vec{q}\|}\left(\alpha \cdot \mathbf{X}\right)\right)\left(v_1, v_2\right) = \left(\mathbf{gvcd}_{\|\vec{q}\|}\left(\mathbf{X}\right)\right)\left(\frac{1}{\alpha} v_1, \frac{1}{\alpha} v_2\right)$
$\vec{f}(\mathbf{X})$	$\left(\mathbf{gvcd}_{\|\vec{q}\|}\left(\vec{f}\left(\mathbf{X}\right)\right)\right)\left(v_1, v_2\right) = \left(\mathbf{gvcd}_{\|\vec{q}\|}\left(\mathbf{X}\right)\right)\left(f^{-1}\left(v_1\right), f^{-1}\left(v_2\right)\right)$
$\mathbf{X} + \mathbf{Y}$	keine generelle Aussage möglich!
$\mathbf{X} \cdot \mathbf{Y}$	keine generelleAussage möglich!

Wir suchen also eine Abbildung, die die Grauwerte des Originalsignals derart transformiert, dass die daraus zu gewinnenden Invarianten über die vorgestellte Integration, bezüglich der Erkennungsleistung durch eine Support-Vektor-Maschine (SVM), optimal sind. Diese Problemstellung kann noch erweitert werden und zwar auf die Suche nach zwei unabhängigen Grauwerttransformationen f_1 und f_2, die jeweils für eine Dimension der GVCD verantwortlich sind.

Die Korrespondenztabelle geht nun von invertierbaren Grauwerttransformationen aus. Dies ist nicht zwangsläufig notwendig, da es zum Beispiel auch sinnvoll sein kann, einzelne Werte der Originalsignale überhaupt nicht zu berücksichtigen und sie deshalb auf einen gemeinsamen Grauwert abzubilden, z.B. die 0. Diese intuitiven Ideen und Betrachtungen führen zur Kernidee der Wahl der Kernfunktion:

Wenn eine Gewichtung der Einträge der GVCD derart existiert, dass für die Klassifikation wichtige Einträge ein hohes Gewicht erhalten und unwichtige ein niedriges, so wird das Integral von denen mit hohem Gewicht und damit den für die Klassifikation entscheidenden dominiert. Die bezüglich der späteren Erkennung optimale Gewichtung ist die optimale Kernfunktion.

Dies gibt nun nur die Richtung vor, es bleibt noch eine Reihe von Fragen zu klären:

- Welches Maß soll verwendet werden um die Wichtigkeit eines Grauwertpaares zu messen?
- Wie wird die Qualität der Gewichtung bewertet?
- Wie lassen sich aus den Gewichten der einzelnen Einträge Grauwerttransformationen f_1 und f_2 gewinnen, die die optimale Kernfunktion möglichst gut approximieren.

Auf den letzten Punkt wird im nächsten Abschnitt eingegangen, die anderen folgen im weiteren Verlauf dieses Kapitels.

4.3. Übertragung der Kernfunktionen auf die Originaldaten

Annahme: Die optimale Kernfunktion $f(v_1, v_2)$ ist auf einem kleinen Datensatz bestimmt. Nun erhält man weitere Daten des gleichen Typs und möchte ohne Berechnung der GVCD die mit der Kernfunktion gewichteten 2-Punkt-Invarianten bestimmen.

Dies ist in einem Schritt nur möglich, wenn die Kernfunktion separierbar ist, was im Allgemeinen nicht der Fall sein wird. Von einer nicht separierbaren Kernfunktion ausgehend ist die Berechnung der gewichteten Invariante etwas aufwendiger. Ein verlustfreies Verfahren ergibt sich aus einer teilweisen Berechnung der GVCD. Um die Idee zu vermitteln, wird die Berechnung der Invariante unter Kenntnis der GVCD genauer analysiert.

Wird für die Berechnung der GVCD die Korrelationsmethode verwendet, ergibt sich die 2-Punkt-Invariante zu:

$$T\left[f\right]\left(\mathbf{X}\right) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f\left(v_1, v_2\right) \cdot \int_{-\vec{\infty}}^{\vec{\infty}} \mathbf{C}_{||\vec{q}||}\left(\vec{x}\right) \cdot \left(\mathbf{M}_{v_1} \otimes \mathbf{M}_{v_2}\right)\left(\vec{x}\right)\ \mathrm{d}\vec{x}\ \mathrm{d}v_1\ \mathrm{d}v_2 \tag{4.1}$$

Für eine genauere Analyse wird die Maskierungsfunktion wieder explizit eingesetzt und die Korrelation in ihrer Integralform ausgeschrieben:

$$T\left[f\right]\left(\mathbf{X}\right) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f\left(v_1, v_2\right) \cdot \int_{-\vec{\infty}}^{\vec{\infty}} \mathbf{C}_{||\vec{q}||}\left(\vec{x}\right) \cdot \int_{-\vec{\infty}}^{\vec{\infty}} \delta\left(v_1 - \mathbf{X}\left(\vec{\tau}\right)\right) \cdot \delta\left(v_2 - \mathbf{X}\left(\vec{\mathbf{x}} + \vec{\tau}\right)\right)\ \mathrm{d}\vec{\tau}\ \mathrm{d}\vec{x}\ \mathrm{d}v_1\ \mathrm{d}v_2$$

Hält man nun einen Grauwert fest und zieht die Kernfunktion, die dann nur noch eine eindimensionale Funktion des verbleibenden Grauwertes ist, in das innere Integral, erhält man die Auswertung des Integrals über eine Grauwertdimension.

$$T\left[f\right]\left(\mathbf{X}\right)\left(v_2\right) = \int_{-\infty}^{\infty}\int_{-\vec{\infty}}^{\vec{\infty}}\int_{-\vec{\infty}}^{\vec{\infty}} \mathbf{C}_{||\vec{q}||}\left(\vec{x}\right) \cdot f\left(v_1, v_2\right) \cdot \delta\left(v_1 - \mathbf{X}\left(\vec{\tau}\right)\right) \cdot \delta\left(v_2 - \mathbf{X}\left(\vec{\mathbf{x}} + \vec{\tau}\right)\right)\ \mathrm{d}\vec{\tau}\ \mathrm{d}\vec{x}\ \mathrm{d}v_1$$

Eine Vertauschung der Integrationsreihenfolge mit anschließendem geschickten herausziehen der vom Grauwert v_1 unabhängigen Terme, führt zu folgender Umformulierung:

$$T\left[f\right]\left(\mathbf{X}\right)\left(v_2\right) = \int_{-\vec{\infty}}^{\vec{\infty}}\int_{-\vec{\infty}}^{\vec{\infty}} \mathbf{C}_{||\vec{q}||}\left(\vec{x}\right) \cdot \delta\left(v_2 - \mathbf{X}\left(\vec{\mathbf{x}} + \vec{\tau}\right)\right) \cdot \underbrace{\int_{-\infty}^{\infty} f\left(v_1, v_2\right) \cdot \delta\left(v_1 - \mathbf{X}\left(\vec{\tau}\right)\right)\ \mathrm{d}v_1}_{= f\left(\mathbf{X}\left(\vec{\tau}\right), v_2\right)}\ \mathrm{d}\vec{\tau}\ \mathrm{d}\vec{x}$$

Das innere Integral kann durch eine einfache Grauwertsubstitution des unmaskierten Signals mit den korrespondierenden Werten der Kernfunktion $f\left(v_1, v_2\right)$ aufgelöst werden.

Das Integral über die verbleibende Grauwertdimension führt nun wieder zur 2-Punkt-Invariante, die auch durch eine vollständige Berechnung der GVCD bestimmt worden wäre. Der Vorteil ist, dass, anstatt über beide Grauwertdimensionen zu integrieren, das Integral über eine Grauwertdimension ausreicht. Die vollständige Berechnungsformel zur Berechnung der Invariante auf diesem Weg ist in Gleichung 4.2 notiert.

$$T\left[f\right]\left(\mathbf{X}\right) = \int_{-\infty}^{\infty}\int_{-\vec{\infty}}^{\vec{\infty}}\int_{-\vec{\infty}}^{\vec{\infty}} \mathbf{C}_{\|\vec{q}\|}\left(\vec{x}\right)\cdot\delta\left(v_2 - \mathbf{X}\left(\vec{\mathbf{x}}+\vec{\tau}\right)\right)\cdot f\left(\mathbf{X}\left(\vec{\tau}\right), v_2\right)\,\mathrm{d}\vec{\tau}\,\mathrm{d}\vec{x}\,\mathrm{d}v_2 \tag{4.2}$$

Diese Berechnungvorschrift ist noch immer relativ aufwendig, da nach wie vor über eine Grauwertdimension integriert werden muss. Eine etwas weniger rechenintensive Idee, ist eine Approximation der Invariante, indem die Kernfunktion durch eine Summe von separierbaren Funktionen angenähert wird, wie zum Beispiel durch Gaußförmige Teilapproximationen. Eine Umsetzung dieser Idee war leider aus Zeitmangel nicht mehr möglich. Auch hat sich gezeigt, dass die getesteten Verfahren zur Bestimmung der Kernfunktionen bereits ohne die Approximation nicht die hohen Erwartungen erfüllten, was in Kapitel 5 näher erläutert wird.

4.4. Feature Weighing und Feature Selection

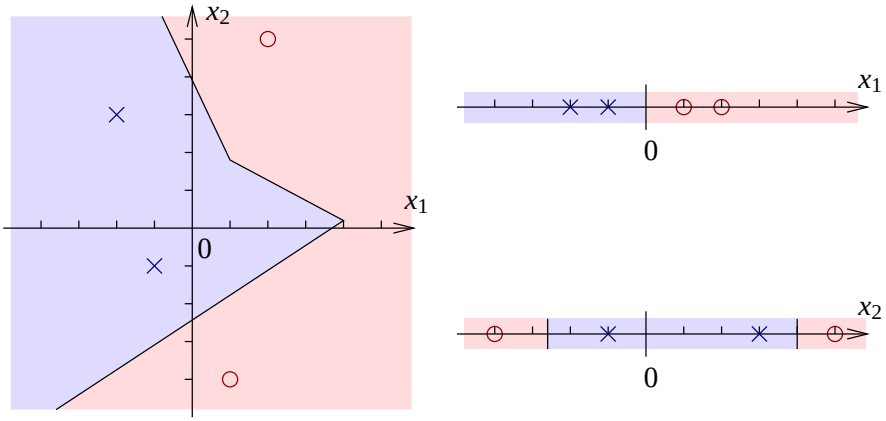
Die Bewertung der diskriminativen Eigenschaften der Merkmale (Features) eines Datensatzes ist insbesondere im Maschinellen Lernen und Data Mining von enormer Bedeutung, da gerade im letztgenannten Gebiet grosse Mengen von Daten möglichst ohne Informationsverlust aber möglichst hoher Informationsdichte gespeichert werden müssen. Daher ist die Merkmalsgewichtung (Feature Weighing), wie auch der Spezialfall der Merkmalsauswahl (Feature Selection), dort beheimatet und man findet eine Fülle von Ideen gute Merkmale zu extrahieren.
Die Datenkompression ist aber nur ein Grund, warum es sinnvoll ist, nur Teilmengen der gegebenen Merkmale zu verwenden. Ein wichtiger weiterer Grund ist der Aufwand bei der Bestimmung eines Modells, das die Merkmalsvektoren auf ihre Klassen abbildet. Ein bekanntes Problem beim Klassifikatorentwurf und damit dem Finden eines obigen Modells ist der so genannte „Curse of Dimensionality": Je höherdimensional die Merkmalsvektoren werden, desto aufwendiger ist es für eine Vielzahl an Trainingsalgorithmen ein entsprechendes Modell zu finden, das auch noch eine gewisse Generalisierungsfähigkeit besitzt, also auch unbekannte Merkmalsvektoren richtig klassifiziert. Um im Allgemeinen eine Generalisierungsfähigkeit zu erreichen, enthält jeder induktive Algorithmus (Klassifikator) einen induktiven BIAS, der a-priori die Eigenschaften des Modells festlegt und damit den Raum aller möglichen Modelle einschränkt. Ein bekannter BIAS, ist „Occam's Razor". Er besagt, dass wenn zwei unterschiedlich komplexe Beschreibungen von Modellen mit gleicher Genauigkeit auf den gegebenen Daten existieren, bevorzuge die einfachere. Diese Heuristik hat in vielen Anwendungsbeispielen zu guten Ergebnissen geführt, weil effektiv ein Overfitting des Klassifikators an die Trainingsbeispiele verhindert werden kann. Je nach Klassifikationsalgorithmus können für die Klassifikation irrelevante Merkmale sogar das Finden eines guten Modells verhindern, wenn fälschlicherweise auf Basis der irrelevanten Merkmale das Modell bestimmt wird, oder der Einfluss dieser zu groß ist. Dies wird insbesondere zum Problem, wenn nur wenige Trainingsbeispiele zur Verfügung stehen. Für nähere Informationen über die beschriebenen Effekte, sowie die generelle Bedeutung des induktiven BIAS, verweise ich auf das Buch „Machine Learning" von Tom Mitchell [Mit97], in dem diese detailliert erläutert werden.

Als einfaches Beispiel für ein Anpassen des Klassifikators an irrelevante Daten möchte ich den Nearest Neighbor Klassifikator heranziehen. Der konstruierte Datensatz besteht aus jeweils zwei Merkmalen, wobei

die Klassenzugehörigkeit ausschließlich vom ersten Merkmal abhängt und es gilt $y = \text{sign}\left(x_1\right)$. Das zweite Merkmal ist zufällig gleichverteilt aus dem Bereich $[-1, +1]$ gewählt und hat damit für die Klassenzugehörigkeit keine Bedeutung:

Beispiel 4.4.1:
Klassifikator: Nearest Neighbor
gelabelter Trainingsdatensatz:

$$S = \left\{ \left(\begin{pmatrix} 0.1 \\ -0.4 \end{pmatrix}, 1 \right), \left(\begin{pmatrix} 0.2 \\ 0.5 \end{pmatrix}, 1 \right), \left(\begin{pmatrix} -0.2 \\ 0.3 \end{pmatrix}, -1 \right), \left(\begin{pmatrix} -0.1 \\ -0.1 \end{pmatrix}, -1 \right) \right\}$$



Der Einfluss, des zweiten Merkmals ist bei Betrachtung beider Merkmale durch den Klassifikator so groß, dass nur in einer sehr kleinen Umgebung das gesuchte Konzept $y = \text{sign}\left(x_1\right)$ gelernt wird. Bei ausschließlicher Betrachtung des ersten Merkmals wäre in diesem Fall die korrekte Hypothese bestimmt worden. Die Wahl nur des zweiten Merkmales wäre fatal gewesen, da dadurch komplexe Klassifikationsgebiete (in 1D mehrere Intervalle) entstanden wären, die mit der gesuchten Funktion nichts mehr gemeinsam hätten.

Man sieht, für die Klassifikation wichtige Merkmale können durch irrelevante an Gewicht verlieren, man sieht außerdem, dass die Wahl der falschen Merkmale den Klassifikatorentwurf erschweren kann, wenn nicht gar unmöglich macht. Dies war natürlich ein konstruiertes Beispiel und man kann argumentieren, dass solche Fälle so nie auftreten, aber in der Tat, kann zum Beispiel ein Sensorrauschen bei ungeschickter Wahl der Merkmale zu einem solchen Fall führen. Um dies zu verhindern, gibt es nun verschiedene Möglichkeiten: frühe Elimination durch Vorverarbeitung oder späte Elimination, bzw. Gewichtung nach einer Merkmalsanalyse.

4.4.1. Einordnung der Algorithmen zur Merkmalsgewinnung

Es gibt verschiedene Aspekte, nach denen man bekannte Verfahren einordnen kann:

- Feature Selection vs. Feature Extraction
  Es gibt zwei Möglichkeiten aus einer Menge von Merkmalen unterschiedlicher Qualität eine kleinere Menge hochwertiger Merkmale zu gewinnen. Die eine ist die Merkmals Selektion, bei der einfach eine Teilmenge der vorhandenen Merkmale geeignet bewertet und die Merkmale mit bester Bewertung ausgewählt werden. Die Merkmals-Extraktion ist etwas komplizierter. Sie ist im Allgemeinen eine Transformation des ursprünglichen Merkmalsraumes in einen anderen Merkmalsraum mit den

gewünschten Eigenschaften. Dies ist sehr allgemein gehalten, aber beschreibt genau das zugrundeliegende Prinzip. Als Beispiel möchte ich die Karhunen Loève Transformation (KLT) nennen. Der Originalraum, wird dabei einer einfachen Basistransformation unterzogen. Die Basisvektoren der Transformierten sind die nach ihren Eigenwerten absteigend sortierten Eigenvektoren der Korrelationsmatrix (oder auch der Kovarianzmatrix, je nach Definition) der Merkmalsvektoren. Die Transformierte zeichnet sich durch eine Dekorrelation der Merkmalsvektoren aus. Aus der KLT lässt sich nun aber nicht direkt ein Selektionsalgorithmus gewinnen, sondern man erhält Gewichte für jedes einzelne Merkmal, aus denen man die Hauptachsen der Verteilung der Merkmalsvektoren ablesen kann. Wie mit Hilfe der KLT auch eine Merkmalsgewichtung mit anschließender Selektion möglich ist wird im nächsten Abschnitt vorgestellt.

- Filter Methods vs. Wrapper Methods
  Die Filter Methods machen sich statistische Aussagen über den Informationsgehalt der Merkmale zu nutze, wie z.B. der Information Gain bei Wahl des entsprechenden Features, wie er auch zum Aufbau von binären Entscheidungsbäumen im ID3 Algorithmus verwendet wird. In [Fle04] wird auf dieser Idee aufbauend ein sehr schneller Feature Selection Algorithmus vorgestellt, der allerdings leider nur auf binären Merkmalen Anwendung findet. Der große Vorteil der Filter Methods ist ihre Unabhängigkeit vom für die Klassifikation verwendeten Klassifikator (induktiver Lerner).
  Im Kontrast dazu stehen die Wrapper Methods, die gewählte Teilmengen der Merkmale direkt mit dem Klassifikator bewerten. Diese Methode garantiert das Finden der optimalen Teilmenge der Merkmale wenn alle 2^n Teilmengen bei n Merkmalen betrachtet werden. Dieses Vorgehen ist jedoch bei der in dieser Arbeit auftretenden Problemstellung, bei der einige tausende Merkmale zu bewerten sind impraktikabel, insbesondere, da der gewählte Klassifikator eine Supportvektormaschine ist, bei der die Berechnung eines Modells auf den gegebenen Daten im Vergleich zu den häufig in der Literatur referenzierten k-Nearest Neighbor Verfahren einen nicht unwesentlichen Aufwand erfordert.

- Discrete/Binary Features vs. Continuous Features
  Bereits im vorhergehenden Punkt wurde ein Algorithmus erwähnt, der auf binären Merkmalen sehr schnell diskriminative Teilmengen selektieren kann. Auf Basis von binären Merkmalsvektoren gibt es sehr viele Ansätze, die sich mit etwas Aufwand auch auf diskrete Werte der Merkmale erweitern lassen, wie zum Beispiel die bereits erwähnten Entscheidungsbäume, genetische Algorithmen und andere meist auf statistischen Betrachtungen beruhende Verfahren. In unserem Fall sind die Merkmale jedoch kontinuierlich. Eine Möglichkeit die bekannten binären Verfahren auf diese Merkmale anzuwenden, ist eine Quantisierung mit anschließender binärer Codierung. Der große Nachteil dieser Idee ist die enorme Erhöhung der Dimension des Merkmalsraumes. Als Beispiel betrachte man eine Quantisierung einer Fließkommazahl in 10 Intervalle. Jedes Intervall wird durch ein Bit codiert. Der resultierende Merkmalsvektor ist nun 10mal so lang wie der ursprüngliche Merkmalsvektor und enthält durch die sehr grobe Quantisierung nur noch einen Bruchteil der ursprünglichen Information.

- Two-Class vs. Multi-Class
  Wie bei der Klassifikation ist auch bei der Aussagekraft der Merkmale die Anzahl der zu unterscheidenden Klassen von Bedeutung. Verschiedene Merkmale haben meist bezüglich einer Klasse einen hohen Informationsgehalt oder unterscheiden zwischen 2 Klassen. Dies führt zu unterschiedlichen Ansätzen: Wie bei der Multiklassen-SVM kann hier eine One vs. Rest, bzw. eine One vs. One Strategie verfolgt werden. Dies bedeutet, dass jedes Merkmal nicht nur ein absolutes Gewicht erhält, sondern die Gewichte je nach betrachteter Klasse, bzw. betrachtetem Klassenpaar variieren können.

- Forward vs. Backward selection
  Die Selektionsrichtung entscheidet bei vielen Verfahren mit über die Qualität der ausgewählten Merkmale. Man unterscheidet Vorwärts Selektion, bei der von der leeren Merkmalsmenge ausgehend so lange Merkmale hinzugenommen werden, bis die Erkennungsleistung ihr Maximum erreicht hat und

Rückwärts Selektion, bei der aus der vollständigen Merkmalsmenge irrelevante Merkmale entfernt werden, solange die Erkennungsleistung nicht abfällt.

In unserem Fall haben wir sehr viele kontinuierliche Merkmale, von denen mit hoher Wahrscheinlichkeit ein Großteil für die Klassifikation bedeutungslos ist. Die Relevanz eines Merkmales hängt von zwei Faktoren ab:

Individueller Informationsgehalt Dieser lässt sich für jedes Merkmal unabhängig von den anderen berechnen und spiegelt die Korrelation zwischen den Werten des Merkmals und der Klassenzugehörigkeit des durch das Merkmal beschriebenen Objektes wieder. Maximale Korrelation besteht dabei, wenn es eine direkte Abbildung der Werte des Merkmales auf die Klassenzugehörigkeit gibt.

Unabhängigkeit Der Informationsgehalt allein sagt noch nicht, ob ein individuelles Merkmal für die Klassifikation benötigt wird. Lässt sich alle Information eines Merkmales x_i durch eine Menge anderer bereits betrachteter Merkmale $\{x_1, \ldots, x_k\}$ gewinnen, so trägt x_i keine Information gegeben $\{x_1, \ldots, x_k\}$. Ein einfaches Beispiel ist das mehrfache Vorkommen des gleichen Merkmales.

4.4.2. Verfahren zur Merkmalsextraktion

Merkmalsextraktion unterscheidet sich, wie bereits angedeutet von der Merkmalsselektion und -gewichtung durch eine etwas andere Grundidee. Anstatt die vorhandenen Merkmale zu bewerten, wird der Merkmalsraum über eine geschickte Transformation vorverarbeitet. Klassische Vorverarbeitungen sind dabei Transformationen, die auf unterschiedlichste Arten eine neue Beschreibung des Signals erzeugen, wie z.B. Basistransformationen, Linearkombinationen der Merkmale, nichtlineare Kombinationen, Integrale, Differentiale, ...

Der Phantasie sind keine Grenzen gesetzt und man kann sehr spezielle auf das Problem zugeschnittene Merkmale extrahieren, was hier jedoch nicht das Ziel ist. Da alle entwickelten Verfahren unabhängig von einer speziellen Problemstellung sein sollen, kommen nur Verfahren in Frage, die im Allgemeinen zu guten Merkmalen führen können.

4.4.3. Verfahren zur Merkmalsgewichtung und Selektion

Nach dem groben Überblick über die Möglichkeiten der Einordnung der Verfahren zur Merkmalsgewichtung und -selektion, möchte ich nun einige bekannte Verfahren aufgreifen und in Hinblick auf die Verwendbarkeit auf das in dieser Arbeit bestehende Problem hin untersuchen.

Wrapper-Methods

Der naive Wrapper Beispiele für die oben genannten Wrapper-Methods sind zunächst der naive Wrapper, der für den gewählten Klassifikationsalgorithmus die optimale Teilmenge der Merkmale bestimmen kann. Die Idee besteht darin, den Klassifikator auf alle 2^n Teilmengen der n-elementigen Merkmalsmenge anzuwenden und die Teilmenge zu wählen, die das beste Klassifikationsergebnis gebracht hat. Wie unschwer zu erkennen ist, ist die Laufzeit des Algorithmus zum einen exponentiell in der Anzahl Merkmale und erfordert zudem für jede Merkmalsteilmenge eine teure Berechnung eines Klassifikators in diesem Fall mit einer SVM und damit in $O\left(n^3\right)$.

unguided Hillclimbing/Beamsearch Statt alle Teilmengen zu betrachten, gehören diese Algorithmen zum Typ der Forward-Selection Algorithmen. Sie beginnen mit einer leeren Teilmenge aller Merkmale und fügen sukzessive Merkmale hinzu, die die Klassifikationsleistung maximal erhöhen. Dies ist ein Greedy Ansatz, der die Optimalität der gefundenen Merkmalsteilmenge nicht mehr garantiert,

aber „nur" noch $\sum_{i=1}^{k} n - k$ Klassifikatoren lernen muss, um eine Teilmenge von k Merkmalen aus n zu selektieren.

Beide oben beschriebenen Methoden sind für das in dieser Arbeit gestellte Problem ungeeignet, da die GVCD sehr viele Merkmale enthält und der Trainingsalgorithmus um ein SVM-Modell zu lernen zu langsam ist um in realistischer Zeit vernünftige Merkmalsteilmengen durch iterative Anwendung desselben zu erreichen.

Filter-Methods

Conditional Entropy Ein vielversprechender Ansatz, möglichst einfach und unabhängig vom verwendeten Klassifikator eine Gewichtung und Selektion von Merkmalen vorzunehmen, ist ihr individueller Informationsgehalt. Eine Idee diesen Informationsgehalt zu messen, ist die Bestimmung der bedingten Entropie, des Trainingsdatensatzes gegeben ein Merkmal Y. Sei X eine Zufallsvariable mit Werten aus $\Omega = \{1, \ldots, K\}$, so dass $x_i \in X$ gerade die Klasse des i-ten Datensatzes ist, sei weiterhin Y^j eine Zufallsvariable, so dass $y_i^j \in Y^j$ den Wert des j-ten Merkmals des i-ten Datensatzes enthält, dann gilt:

$$\text{Entropy: } H\left(X\right) = -\sum_{\omega \in \Omega} p_\omega \cdot \log p_\omega \tag{4.3}$$

$$\text{Conditional Entropy: } H\left(X|Y^j\right) = \sum_{y^j \in Y^j} p\left(y^j\right) \cdot H\left(X|Y^j = y^j\right) \tag{4.4}$$

Dieses Verfahren ist für Merkmale mit wenigen diskreten Werten sehr gut anwendbar. Insbesondere im binären Fall liefert es hervorragende Ergebnisse. Dieser Spezialfall wurde von J. R. Quinlan untersucht [Qui86]. Leider treten aber bei der Anwendung auf die kontinuierlichen Merkmale dieser Arbeit wieder die zuvor erwähnten Probleme der Quantisierung auf.

Perceptron Weighing Die Idee dieses Verfahrens ist die Beobachtung, dass die Gewichte eines einzelnen Perceptrons stark mit dem Informationsgehalt der eingehenden Signale korrelieren. Alternativ zum Perceptron ist es nun auch möglich eine lineare SVM zu trainieren und die Einträge des Normalenvektors zu verwenden, die dieselbe Eigenschaft haben sollten (vgl. [HCS01]). Diese Form der Gewichtung eignet sich für die Problemstellung dieser Arbeit, im Gegensatz zu den bisher beschriebenen Verfahren, hervorragend und wird im nächsten Abschnitt näher beschrieben.

Principal Feature Analysis Eine Anwendung der Karhunen Loève Transformation zur Feature Selection ist die Principal Feature Analysis (vgl. [CTZH02]). Die Grundlage dieses Verfahrens ist die Beobachtung, dass die Spalten der bei der KLT bestimmten Transformationsmatrix auf die einzelnen Merkmale wirken. Betrachtet man nun nur die Merkmale, deren korrespondierende Transformationsspalten eine hohe Norm aufweisen, erhält man Merkmale mit hoher Diskriminierfähigkeit. Im erwähnten Artikel, wird als zweiter Verarbeitungsschritt eine Clusteranalyse der selektierten Merkmale angestellt, um Merkmale mit ähnlichem Informationsgehalt auf ihr mittleres Merkmal abzubilden. Dies wird durch Anwendung des k-means Algorithmus erreicht und eliminiert effektiv Redundanzen zwischen den Merkmalen.

Neben den eingeführten Verfahren gibt es noch eine Vielzahl weiterer im Laufe der letzten Jahrzehnte entwickelte Algorithmen und Modifikationen, die auf spezielle Probleme zugeschnitten sind und oft nur Variationen darstellen. Eine vollständige Auflistung mit entsprechender Analyse würde den Rahmen dieser Arbeit jedoch sprengen, weswegen auf weitere Algorithmen verzichtet wurde.

In dieser Arbeit wurden zwei Verfahren der Merkmalsextraktion verwendet:

- Perceptrongewichtung
- Transformation des Merkmalsraumes über die Karhunen Loève Transformation

4.5. Feature Gewichtung mit linearer SVM (Perceptrongewichtung)

Eine lineare SVM berechnet die trennende Hyperebene mit maximalem Rand zwischen den Klassen. Betrachten wir also einen Merkmalsraum der Dimension n, so liefert uns eine lineare SVM zwei Vektoren der gleichen Dimension: $\vec{w}$ die Normale der Hyperebene und einen Versatzvektor $\vec{b}$. Für eine lineare SVM im $\mathbb{R}^2$ ist dies in Abb. 4.2 dargestellt: Der blau dargestellte Vektor ist dabei die Normale der Hyperebene

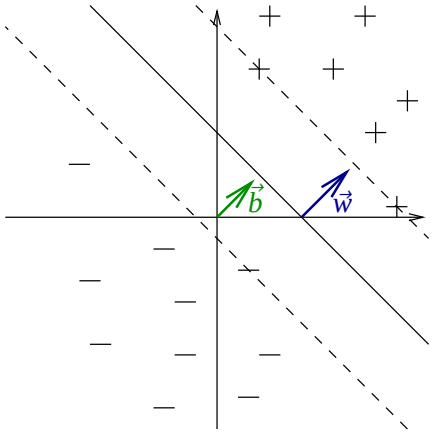


Abbildung 4.2.: Beispiel einer linearen SVM (vollständig separabler Fall)

(auf 1 normiert), der grüne der Versatz. Ist nun ein Merkmal irrelevant für die Klassifikation, werden die Werte, die dieses Merkmal in den Beispielen annimmt unabhängig von der Klasse beliebige Werte haben. Das bedeutet für die Normale, dass sie in Richtung dieses Merkmales nur relativ kleine Beträge aufweisen wird. Bei gut diskriminierenden Features wird dieser Betrag erwartungsgemäß größer ausfallen, so dass eine direkte Gewichtung der Features mit den Beträgen der entsprechenden Einträge des Normalenvektors, eine gute Heuristik darstellt.

Hat man nun mehr als zwei Klassen zu unterscheiden, kann man zwei Strategien verfolgen:

One-vs-One Für jedes Klassenpaar, wird eine SVM trainiert. Merkmalsvektoren anderer Klassen bleiben dabei für das Training unberücksichtigt. Als Ausgabe erhält man bei K Klassen $\frac{K(K+1)}{2}$ lineare SVM-Modelle, die die jeweils optimal trennende Hyperebene für jedes Klassenpaar beschreiben.

One-vs-Rest Hier wird für jede Klasse nur ein Modell berechnet, für das alle anderen Klassen zu einer gemeinsamen Rest-Klasse kombiniert werden. Die Ausgabe sind hier die optimal trennenden Hyperebenen, die die betrachtete Klasse am besten von den restlichen trennt. Die Wahrscheinlichkeit, dass es überhaupt eine Hyperebene gibt, die die Trainingsbeispiele optimal trennen kann, ist in diesem Fall wesentlich geringer als im One-vs-One Fall. Damit, und mit der wesentlich größeren Zahl an zu betrachtenden Trainingsbeispielen pro SVM Training macht diese Strategie aufwendiger, was sich in einer erheblich längeren Trainingszeit bemerkbar macht. Je nach Normierung der Daten mussten solche Berechnungen im Laufe dieser Arbeit aufgegeben werden.

Für jede gelernte SVM können nun die Merkmale mit den Normalenvektoren der Hyperebenen gewichtet werden, und man erhält eine Reihe von Merkmalen, die einzeln nur sehr wenig über das vollständige Problem mit K Klassen aussagen, jedoch in Kombination zu einer hohen Diskriminierfähigkeit führen sollten.

Eine detaillierte Beschreibung der, der SVM zugrundeliegenden Mathematik ist [Bur98] zu entnehmen.

4.6. Feature Gewichtung mit Hilfe der Karhunen Loève Transformation

Als Beispiel für eine Methode der Merkmalsextraktion mit Hilfe von Transformationen wurde die Karhunen Loève Transformation (KLT) gewählt, da sie besondere Eigenschaften hat, die sie für das hier gestellte Problem auszeichnet. Bevor ich jedoch auf diese Eigenschaften eingehe, möchte ich kurz auf die Hintergründe und die Berechnung der KLT eingehen.

4.6.1. Berechnung der KLT

Satz 4.6.1: *Schätzung der Parameter eines stochastischen Prozesses*
Sei $\mathbf{X}$ ein stochastischer Prozess mit Zufallsereignissen $\vec{x}^j$, wobei jedes Zufallsereignis durch eine geordnete Menge von Elementarereignissen $\vec{x}^j = (x_1^j, x_2^j, \ldots, x_n^j)^T$ beschrieben wird, dann lassen sich Erwartungswert $\vec{\mu}_{\mathbf{X}}$ und Autokorrelationsmatrix $R_{\mathbf{XX}}$ des Prozesses folgendermaßen anhand von Stichproben schätzen:

$$\vec{\mu}_{\mathbf{X}} = \mathrm{E}\left(\vec{x}\right) \approx \frac{1}{N}\sum_{i=1}^{N}\vec{x}^i$$

$$\mathbf{R_{XX}} = \mathrm{E}\left\{\vec{x}\vec{x}^T\right\} = \left\{\mathrm{E}\left(x_i x_j\right)\right\} \approx \frac{1}{N}\sum_{i=1}^{N}\begin{pmatrix} x_0^i x_0^i & x_0^i x_1^i & \cdots & x_0^i x_N^i \\ x_1^i x_0^i & x_1^i x_1^i & \ldots & x_1^i x_N^i \\ \vdots & \vdots & \ddots & \vdots \\ x_N^i x_0^i & x_N^i x_1^i & \cdots & x_N^i x_N^i \end{pmatrix}$$

Auf Basis der Autokorrelationsmatrix lässt sich nun die KLT folgendermaßen definieren:

Definition 4.6.1: *Karhunen Loève Transformation*
Sei $\mathbf{R_{XX}}$ die Autokorrelationsmatrix eines stochastischen Prozesses. Sei weiterhin $\mathbf{A} = \left(e'_1, e'_2, \ldots, e'_N\right)$ die Matrix der Eigenvektoren von $\mathbf{R_{XX}}$ und $\mathbf{\Lambda} = \mathrm{diag}\left(\lambda_1, \lambda_2, \ldots, \lambda_N\right)$ die Diagonalmatrix, der korrespondierenden Eigenwerte λ_i, so dass $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$ und gelte weiterhin:

$$\begin{aligned} \mathbf{A}^T\mathbf{A} &= \mathbf{I} \\ \mathbf{A}^T\mathbf{R_{XX}}\mathbf{A} &= \mathbf{\Lambda} \end{aligned}$$

Dann ist die KLT folgendermaßen definiert:

$$\mathrm{KLT}\left(\vec{x}\right) = \vec{y} := \mathbf{A}^T\vec{x}$$

Die inverse Transformation ergibt sich dann zu:

$$\mathrm{KLT}^{-1}\left(\vec{y}\right) = \vec{x} = \mathbf{A}\vec{y}$$

4.6.2. Eigenschaften der KLT

- Die transformierten Vektoren $\vec{y}$ sind unkorreliert, da ihre Korrelationsmatrix $\mathbf{R_{YY}}$ eine Diagonalmatrix ist (Dekorellation)
- Die Varianz der Transformierten fällt monoton mit aufsteigenden Koeffizienten, womit nach einer festen, meist kleinen Anzahl von Basisvektoren bereits nahezu die vollständige Information kodiert werden kann (Informationsverdichtung)
- Die Varianzen der transformierten Merkmale sind maximal ungleichgewichtig (Entropieminimierung)

Der einzige Nachteil ist, dass eine auf alle Daten angewendete KLT keine speziellen Klasseninformationen nutzen kann und daher zwar der Merkmalsraum optimal abgebildet wird bezüglich der generellen Merkmalsinformationen, jedoch nicht bezüglich der Klassenzugehörigkeiten.

Um diese mit einzubeziehen, gibt es nun die Möglichkeit für jede Klasse $K_i \in \{K_1, K_2, \ldots, K_k\}$ die Transformationsmatrix A_i zu bestimmen und anschliessend jeden Merkmalsvektor unabhängig von der Klassenzugehörigkeit mit allen A_i zu transformieren. Dadurch erhält man k-mal mehr transformierte Merkmale, die gruppenweise einer Klasse zugeordnet werden können und deren statistische Verteilung beschreiben.

Wie beschrieben ist es dabei nicht notwendig die vollständige Transformation durchzuführen, sondern man kann sich auf die ersten Basisfunktionen beschränken, was die Merkmalsvektoren wieder etwas handlicher macht.

Zur Berechnung der KLT in den nachfolgenden Experimenten wurde die MATLAB Funktion `princomp` verwendet.

5. Experimente und Ergebnisse

5.1. Merkmalsgeneration

Als Basis für die ersten durchgeführten Versuche zur Merkmalsextraktion wurden auf den beschriebenen 3D- und 2D-Daten GVCDs mit einer Quantisierung von 16 Grauwerten berechnet. Für jeden Datensatz und jede Kanalkombination wurden 39 Parameterkombinationen verwendet:

$$\begin{aligned} P_1 = \{ \quad & (0,1), \ldots, (0,8), \\ & (1,1), \ldots, (1,8), \\ & (2,1), \ldots, (2,8), \\ & (4,1), \ldots, (4,8), \\ & (8,1), \ldots, (8,4), \\ & (16,1), (16,2), \\ & (32,1) \ \} \end{aligned}$$

Dadurch erhält man $16 \times 16 \times 39 = 9984$ Merkmale für jede Kanalkombination, also genau diese Gesamtzahl für die 1-Kanal 3D-Daten und dreimal so viele (29952) für die 2-Kanal 2D-Daten.

Zur Berechnung der GVCDs wurde der GVCD_FAST Algorithmus verwendet.

Wie sich gezeigt hat ist die Normierung der Daten von großer Bedeutung. Um dies zu zeigen wurden drei verschiedene Arten der Normierung implementiert und getestet:

keine Normierung Die GVCD wird als 2-dimensionales Grauwerthistogramm betrachtet und daher keine Normierung angewendet

frühe Normierung Noch vor Berechnung der GVCDs wird das Originalsignal auf eine Grauwertsumme von 1 normiert. Bei Umskalierungen wird keine neue Normierung durchgeführt. Diese Form der Normierung wird in den folgenden Tabellen durch „Signal" abgekürzt.

späte Normierung Nach der Berechnung einer GVCD wird sie auf die Grauwertsumme des zugehörigen skalierten Signals normiert. Dies ist die normale Vorgehensweise zur Berechnung der Standard 2-Punkt-Invarianten In den Tabellen wird diese Normierung durch „GVCD" abgekürzt.

Als Vergleichswerte wurden ausgehend von den GVCDs die Standard 2-Punkt-Grauwert-Invarianten mit Kernfunktion $f(v_1, v_2) = v_1 \cdot v_2$ berechnet.

Um die Klassifikationsleistung der verschiedenen Verfahren zu vergleichen, wurden die jeweils extrahierten Merkmalsvektoren einem 10-fold Cross Validation Test unterzogen. Der verwendete Klassifikator war eine Support Vektor Maschine mit RBF Kern. Die Parameter (C, γ) der SVM wurden in jedem Fall durch wiederholte Cross-Validation-Tests mit verschiedenen Parameterkombinationen gewonnen. Dabei wurden folgende Einzelwerte der Parameter in allen Kombinationen verwendet:

$$\begin{aligned} C &\in \left\{ 2^i \mid i = 0, ..., 26 \right\} \\ \gamma &\in \left\{ 2^i \cdot 10^{-8} \mid i = 0, ..., 26 \right\} \end{aligned}$$

Dadurch ist natürlich nicht die Optimalität dieser Parameter gewährleistet und ein genaueres Parameterfitting kann zu Ergebnissen führen, die die hier präsentierten übertreffen. Da die Parametersuche jedoch sehr zeitraubend ist und nicht einzelne Verfahren bevorzugt behandelt werden sollten, wurde immer die gleiche Parameter-Menge auf den jeweiligen Merkmalsvektoren getestet und das in dieser Menge optimale Parameterpaar ausgewählt.

Auf den vielversprechendsten Ergebnissen aufbauend, wurden weitere GVCDs mit höheren Auflösungen berechnet. Um dies zu ermöglichen wurde anstelle der 8 Punktabstände pro Skalierungsstufe nur noch ein fester Punktabstand von 2px betrachtet. Dies führt zu den Parametern, die bereits in den Veröffentlichungen zu den 2-Punkt-Invarianten verwendet wurden (vgl. [RBS02]). Der neue Parametersatz besteht dann aus folgenden (σ, $\left\|\vec{q}\right\|$)-Paaren:

$$P_{\mathrm{GSI}} = \left\{ (0,2), (1,2), (2,2), (4,2), (8,2), (16,2) \; (32,2) \right\}$$

Da immer ein Punktabstand von 2px verwendet wird, kann der naive Algorithmus verwendet werden, so dass die Grauwertauflösung beliebig angehoben werden kann ohne Erhöhung des Berechnungsaufwandes. Es wurden Grauwertauflösungen von 32 und 64 Grauwerten verwendet. Höhere Auflösungen haben den Rechenaufwand für SVM, wie auch für KLT zu sehr erhöht.

5.2. Perceptrongewichtung

5.2.1. 3D Daten - Parametersatz P_1

Tabelle 5.1 zeigt die Ergebnisse der Cross-Validation Tests auf den gewichteten Merkmalen der 3D Pollen Daten.

Die Berechnung der Hyperebenen-Normalen im One-vs-Rest Fall hat ohne Normierung und mit früher Normierung leider mehrere Wochen gedauert und musste daher abgebrochen werden um Rechenleistung für vielversprechendere Ideen freizugeben, weswegen dazu die Ergebnisse leider nicht zur Verfügung stehen. Im Fall der späten Normierung ist entgegen den Vermutungen die Erkennungsleistung drastisch gesunken.

Auch im One-vs-One Fall erkennt man, dass die Erkennungsleistung wider Erwarten bei zwei der drei gewählten Normierungen gesunken ist, was nur dadurch zu erklären ist, dass die Parameter der linearen SVMs zur Bestimmung der trennenden Hyperebenen schlecht gewählt waren. Es war leider unmöglich über wiederholte Trainings-Test-Läufe die optimalen Parameter zu bestimmen, da bereits für sehr gängige Werte des Bestrafungsterms C, der bei der linearen SVM den größten Einfluss hat, die Laufzeit nicht mehr im Verhältnis zum Nutzen stand.

Kernfunktion $f(v_1, v_2)$	#Merkmale	Normierung	% correct
$v_1 \cdot v_2$	39	—	86.75
		Signal	87.01
		GVCD	86.32
Hyperebenennormalen der 26 linearen One-vs-One SVMs	1014	—	N/A
		Original	N/A
		GVCD	76.88
Hyperebenennormalen der 351 linearen One-vs-One SVMs	13689	—	88.05
		Original	84.94
		GVCD	77.14

Tabelle 5.1.: Cross-Validation Ergebnisse auf der 3D-LSM Datenbank unter Verwendung von Hyperebenennormalen linearer SVMs als Kernfunktion und Parametersatz P_1 für die GVCD Berechnung

5.2.2. 2D Daten - Parametersatz P_1

Tabelle 5.2 zeigt die Ergebnisse der Cross-Validation Tests auf den gewichteten Merkmalsvektoren der 2D-2-Kanal-Pollendaten.

Kernfunktion $f(v_1, v_2)$	#Merkmale	Normierung	% correct
$v_1 \cdot v_2$	117	—	92.19
		Signal	92.14
		GVCD	95.06
Hyperebenennormalen der 12 linearen One-vs-One SVMs	1404	—	91.79
		Original	91.68
		GVCD	88.52
Hyperebenennormalen der 78 linearen One-vs-One SVMs	9126	—	91.85
		Original	91.73
		GVCD	88.17

Tabelle 5.2.: Cross-Validation Ergebnisse auf der 2D-2-Kanal Datenbank unter Verwendung von Hyperebenennormalen linearer SVMs als Kernfunktion und Parametersatz P_1 für die GVCD Berechnung

Die Berechnungen der Hyperebenen ging in diesem Fall zwar schneller, aber auch hier war es unmöglich in realistischer Zeit gute Parameter für die lineare SVM zu finden. Die Erkennungsleistung ist, wie schon im 3D Fall gesehen, schlechter als die Referenzergebnisse. Was hier ins Auge sticht, ist, dass die Referenzinvarianten bei später Normierung eine wesentlich höhere Qualität aufwiesen bezüglich der Erkennungsleistung, als die entsprechenden Invarianten bei anderer Normierung. Dies wäre ansich nicht verwunderlich, aber das absolut gegenteilige Verhalten der gewichteten Merkmale ist ungewöhnlich und ich bin nicht in der Lage dieses Phänomen zu erklären.

Zusammenfassend, muss leider festgestellt werden, dass die intuitive Idee der direkten Gewichtung der Merkmale mit den Hyperebenennormalen auf diesen Daten nicht zu befriedigenden Ergebnissen geführt hat. Ich nehme an, dass eine stärkere Bestrafung von Fehlklassifikationen im Trainingsschritt durch Wahl eines größeren Bestrafungstermes C zu besseren Hyperebenen geführt hätte, allerdings kann ich dies nicht anhand von Ergebnissen untermauern.

5.3. Karhunen Loève Transformation

5.3.1. 3D Daten - Parametersatz P_1

Tabelle 5.3 zeigt die Ergebnisse der Cross-Validation Tests auf den gewichteten Merkmalen der 3D Pollen Daten.

Kernfunktion $f(v_1, v_2)$	#Merkmale	Normierung	% correct
$v_1 \cdot v_2$	39	—	86.75
		Signal	87.01
		GVCD	86.23
Basisvektoren 1-10 der über der gesamten Datenbank berechneten KLT	390	—	89.87
		Original	90.39
		GVCD	75.84
Basisvektoren 1-10 der für jede Klasse separat berechneten KLTs	10140	—	90.91
		Original	91.69
		GVCD	79.33

Tabelle 5.3.: Cross-Validation Ergebnisse auf der 3D-LSM Datenbank unter Verwendung der KLT zur Bestimmung der Kernfunktion und Parametersatz P_1 für die GVCD Berechnung

Die ersten drei Spalten zeigen wie zuvor die Referenzergebnisse der durch die GVCD approximierten 2-Punkt-Invarianten.

Die darauf folgenden Zeilen, zeigen die Ergebnisse, wenn die Basis der KLT auf dem vollständigen Datensatz berechnet wird. Man erkennt trotz der fehlenden Klasseninformation in zwei von drei Fällen eine deutliche Verbesserung in der Erkennungsleistung. Wie schon im Fall der SVM im letzten Abschnitt, ist auch hier zu sehen, dass die späte Normierung die Klassifikation erschwert und in diesem Fall sogar zu noch gravierenderen Verschlechterungen in der Erkennungsleistung führt, als man dies bei der SVM beobachten konnte.

Die letzten drei Zeilen zeigen schließlich die Ergebnisse, die erzielt werden konnten indem für jede Klasse einzeln die KLT Basis bestimmt wurde um dann jeweils die ersten 10 Basisvektoren als Kernfunktionen zu verwenden. Man erkennt hier wie erwartet noch einmal eine Verbesserung gegenüber der klassenunabhängigen KLT. Es muss allerdings bemerkt werden, dass für die Berechnung der KLT immer alle Daten hinzugezogen wurden. Aus Gründen der Rechenzeit war ein vollständiger Leave-One-Out Crossvalidation-Test mit Berechnung der KLTs in jedem Schritt leider nicht möglich, weshalb die Ergebnisse tendenziell etwas besser sein dürften, als dies für völlig unbekannte Daten der Fall wäre.

5.3.2. 2D Daten - Parametersatz P_1

Tabelle 5.4 zeigt die Ergebnisse der Cross-Validation Tests auf den gewichteten Merkmalsvektoren der 2D-2-Kanal-Pollendaten.

Diese Ergebnisse sind sehr interessant, da für den größeren Datensatz die Normierung scheinbar keinen Einfluss mehr auf die Gesamterkennungsleistung hatte. Nach den bisherigen Ergebnissen ist dies überraschend, da insbesondere die spätere Normierung bisher immer zu erheblich schlechteren Ergebnissen geführt hatte als die restlichen. Eine mögliche Erklärung ist die genauere Approximation der echten Merkmalsverteilungen dadurch, dass mehr Datensätze zur Verfügung standen. Was aus diesen Ergebnissen nicht ersichtlich

Kernfunktion $f(v_1, v_2)$	#Merkmale	Normierung	% correct
$v_1 \cdot v_2$	117	—	92.19
		Signal	92.14
		GVCD	95.06
Basisvektoren 1-10 der über der gesamten Datenbank berechneten KLT	1170	—	94.95
		Original	94.95
		GVCD	94.95
Basisvektoren 1-10 der für jede Klasse separat berechneten KLTs	14040	—	95.01
		Original	95.01
		GVCD	95.01

Tabelle 5.4.: Cross-Validation Ergebnisse auf der 2D-2-Kanal Datenbank unter Verwendung der KLT zur Bestimmung der Kernfunktion und Parametersatz P_1 für die GVCD Berechnung

ist, aber erwähnt werden sollte, ist eine sehr große Unempfindlichkeit der berechneten Merkmalsvektoren gegenüber der Parameterwahl der zur Klassifikation verwendeten SVM.

5.3.3. Parametersatz P_2

Die guten Erkennungsleistungen, die durch Verwendung der KLT zu erreichen waren, ließen die Frage aufkommen, ob durch eine höhere Grauwertauflösung eine weitere Verbesserung zu erreichen ist. Aus diesem Grund wurden die im folgenden präsentierten abschließenden Experimente durchgeführt. Es wurde ausschließlich die KLT zur Bestimmung der Kernfunktion verwendet und auf die Normierung bei der Berechnung der GVCD verzichtet, da diese Vorgehensweise mit die besten Ergebnisse erzielt hat. Zu beachten, ist, dass der Parametersatz erheblich kleiner ist, was eine Verschlechterung der Erkennungsleistungen trotz der höheren Grauwertauflösung nicht ausschließt.
Die Ergebnisse der Cross-Validation Tests auf den 3D-LSM, bzw. 2D-2-Kanal Daten ist in Tabellen 5.5 und 5.6 zu sehen.

Kernfunktion	#Graustufen	#Merkmale	% correct
$v_1 \cdot v_2$	32	39	81.82
	64	39	81.30
10 KLT-Basisvektoren - vollständige Datenbank	32	390	93.51
	64	390	92.73
10 KLT-Basisvektoren - klassenweise	32	10140	95.58
	64	10140	94.29

Tabelle 5.5.: Cross-Validation Ergebnisse auf der 3D-LSM Datenbank bei höheren Grauwertauflösungen, KLT-Basisvektoren als Kernfunktion und Parametersatz P_2 für die GVCD Berechnung

Die Ergebnisse aus den Experimenten mit dem reduzierten Parametersatz aber der höheren Grauwertauflösung zu analysieren und zu erklären ist einheitlich nicht möglich. Die Referenzergebnisse zeigen, dass wichtige Parameter entfernt wurden, da die Erkennungsleistung durchgehend gesunken ist. Die höhere Grauwertauflösung konnte dies nicht kompensieren.

Die Ergebnisse der Gewichtung über die KLT der gesamten Datenbanken, zeigt eine enorme Datenabhängigkeit, was die spätere Erkennungsleistung betrifft. Die 3D Daten wurden bei höherer Grauwertauflösung

Kernfunktion	#Graustufen	#Merkmale	% correct
$v_1 \cdot v_2$	32	117	89.95
	64	117	89.95
10 KLT-Basisvektoren -	32	1170	80.48
vollständige Datenbank	64	1170	71.99
10 KLT-Basisvektoren -	32	14040	92.88
klassenweise	64	14040	92.88

Tabelle 5.6.: Cross-Validation Ergebnisse auf der 2D-2-Kanal Datenbank bei höheren Grauwertauflösungen, KLT-Basisvektoren als Kernfunktion und Parametersatz P_2 für die GVCD Berechnung

deutlich besser erkannt, als in den bisherigen Versuchen. Im krassen Gegensatz dazu steht die Erkennungsleistung auf den 2D-Daten, die sehr weit abgesunken ist. Hier sieht man eindeutig, dass die fehlende Klasseninformation auf die Erkennungsleistung erheblichen Einfluss haben kann. Dass dies bisher nicht aufgefallen ist, muss als glücklicher Zufall gewertet werden. Man sieht jedenfalls eindeutig, dass diese Form der Gewichtung nur mit Vorsicht zu verwenden ist und keine hohe Erkennungsrate garantiert.

Die Merkmalsgewichtung, in der die Klassenzugehörigkeiten berücksichtigt wurden, verhalten sich wieder wie erwartet und bringen im 2D-Fall wieder eine solide Erkennungsleistung von 92.88%. Die Ergebnisse auf den 3D-LSM Daten sind sogar noch besser geworden, was jedoch wie bereits zuvor bemerkt durch die sehr geringe Beispielzahl pro Klasse und der Berücksichtigung aller Beispiele für die KLT nicht unbedingt auf weitere unbekannte Daten übertragbar ist.

5.4. Schlussfolgerungen und Ausblick

Durch die extrem große Streuung der erzielten Ergebnisse, sind generelle Aussagen über die Qualität der entwickelten Verfahren nur sehr schwierig zu treffen. Bezüglich der auf Basis der GVCD berechneten Referenzergebnisse, haben die Merkmale, die mit Hilfe der Karhunen Loève Transformation mit Klasseninformation bestimmt wurden, eine hohe Stabilität gegenüber Datennormierung und Art der Daten gezeigt. Die Ergebnisse der Cross-Validation-Tests zeigen, dass die gewählten Merkmalsgewichtungen meist deutlich besser abschnitten, als eine einfache lineare Gewichtung der Grauwerte. Im Vergleich zu anderen Ergebnissen auf den gleichen Daten, lagen die in dieser Arbeit erzielten Erkennungsraten jedoch noch etwas unter den besten erzielten Erkennungsraten, was bei der Wahl optimaler Merkmale nicht der Fall sein dürfte.

Die Ergebnisse der Merkmalsgewichtung mit Hilfe der linearen SVM waren leider wider erwarten deutlich schlechter und blieben weit hinter den Erwartungen zurück. Dies kann an der Wahl der Parameter der SVM liegen, die nicht angepasst werden konnten, weitere Experimente in diese Richtung wären erforderlich um dies zu bestätigen. Die hohen Trainingszeiten der SVM erschweren dies leider erheblich.

Insgesamt ist die Zahl der extrahierten Merkmale sehr gross. Dies widerspricht zwar nicht der Aufgabenstellung, ist aber aus erwähnten Gründen nicht erwünscht, zumal bestehende Ansätze mit einem Bruchteil der Merkmale die gleiche Erkennungsleistung erzielen. Durch die klassenweisen Ansätze ist nun allerdings schon eine Minimalzahl an Merkmalen vorgegeben, die nicht unterschritten werden kann, wenn jede Klasse berücksichtigt werden soll. Auch eine gewisse Anzahl an Punktabständen sollte für eine gute Texturanalyse erhalten bleiben, so dass hier die sieben verwendeten Skalierungen als notwendig angesehen werden können. Bleibt die Anzahl der Koeffizienten der KLT. Hier bestünde die Möglichkeit über eine Diskriminanzanalyse Merkmale einzusparen und möglicherweise sogar die Erkennungsleistung zu erhöhen.

Im Verlauf dieser Arbeit wurden neben den implementierten und getesteten Verfahren der Merkmalsextraktion noch eine Reihe weiterer erwähnt. Eine genauere Analyse dieser bestehenden Verfahren auf ihre Anwendbarkeit in der Mustererkennung wäre wünschenswert und würde wahrscheinlich zu wesentlich akkurateren Merkmalsbewertungen führen, als dies hier erreicht wurde. Die bestehende Implementation des „CMIM Feature Selection Tools" von François Fleuret wurde bereits kurz getestet. Für die Anwendung in der Bildverarbeitung muss es jedoch noch angepasst werden, da die Implementation nicht auf die großen Datenmengen, die dort auftreten ausgelegt ist.

In Bezug auf die Berechnung der GVCD, bleibt die Frage nach einem universellen und schnellen Algorithmus offen. Für die gängigsten Parameterbereiche sind die implementierten Algorithmen jedoch anwendbar. Die Berechnungen auf den gegebenen Datenbanken dauerten auf einem 64Bit Rechner mit zwei 2.4GHz Prozessoren für Parametersatz P_1 auf den 2D-Daten ca. 3 Tage. Die Verarbeitung mit dem naiven Algorithmus und dem reduzierten Parametersatz P_2 war innerhalb von 4 Stunden abgeschlossen. Dabei ist zu berücksichtigen, dass jeweils 2 Berechnungen parallel durchgeführt wurden.

A. Dokumentation der entwickelten Programme

gvcd_GSI

Kurzbeschreibung: Berechnung der GVCD

Syntax:

```
gvcd_GSI  [{-a|--algorithm} <number>]
          [{-c|--channels} <<string>>]
          [{-g|--graylevels} <number>]
          [{-n|--normalize} <number>]
          [{-o|--outfile} <string>]
          {-p|--paras} <string>
          [{-s|--sample_num} <number>]
          [{-v|--verbose} <number>]
          <<infiles>>
```

Parameter:

-a|--algorithm Wählt den Algorithmus aus, der zur Berechnung der GVCDs verwendet werden soll:

- **1 -** Naiver Algorithmus (GVCD_NAIV)
- **2 -** Monte Carlo Schätzung (GVCD_MC)
- **3 -** Schnelle Faltung (GVCD_CONV)
- **4 -** Schnelle Korrelation (GVCD_CORR)
- **5 -** Schnelle Korrelation mit sparse Spheres
- ***6 -** Schnelle Korrelation (GVCD_FAST)
- **7 -** Schnelle Scale Space Invariants (GVCD_SSI)

-c|--channels Wählt die Kanäle, die zur Berechnung herangezogen werden. Die Übergebenen Strings müssen netCDF-Variablen Bezeichnungen entsprechen, die in allen Eingabedateien enthalten sind und deren Variablen die gleiche Ausdehnung haben. Wird dieser Parameter nicht angegeben, versucht das Programm alle in den Eingabedateien enthaltenen Variablen zu verarbeiten

-g|--graylevels Wählt die Anzahl Grauwerte und damit die Auflösung der GVCD. Wird dieser Parameter nicht angegeben, werden die GVCDs mit einer Grauwertauflösung von 8 Grauwerten berechnet

-n|--normalize Wählt die Art der Normalisierung der Daten. Unabhängig von der hier getroffenen Wahl werden sie in jedem Fall vor Berechnung der GVCDs auf ihre Standardabweichung normiert, zusätzlich kann man wählen:

- **0 -** Keine Normalisierung
- ***1 -** Normalisierung auf Norm des skalierten Signals. Die Normierung wird in diesem Fall erst nach Berechnung der GVCD angewandt und wird auf das zur Berechnung verwendete skalierte Signal bezogen

2 - Normalisierung auf Norm des Originalsignals. In diesem Fall wird das Eingabesignal einmal vor Maskierung und Berechnung der GVCDs normiert

-o|--outfile Wählt eine Erweiterung, die vor die Ausgabedateien gesetzt wird, um sie von den Eingabedateien zu unterscheiden. Wird dieser Parameter nicht angegeben, heissen die Ausgabedateien so wie die korrespondierenden Eingabedateien, was insbesondere bedeutet, dass diese, falls sie im aktuellen Verzeichnis sind um die GVCD-Variablen erweitert werden. Das Ausgabeformat bei Angabe des Parameters ist <outfile>_<infile name>

-p|--paras Wählt die netCDF Datei, die die Parameter enthält, für die die GVCD berechnet werden soll. Diese Datei muss zwei Attribute enthalten:

sigma Eine Liste der zu verwendenden Skalierungen

span Eine Liste der zu verwendenden Radien

Beide Listen haben dabei die gleiche Länge und die Kombination der Einträge beschreibt einen Parametersatz. z.B.:

```
sigma= 0 1 0 0
span = 1 1 2 3
```
$\Rightarrow \quad \vec{p} = \left\{ (0,1), (1,1), (0,2), (0,3) \right\}$

-s|--sample_num Wählt für den Monte Carlo Algorithmus die Anzahl der Monte Carlo Tests pro GVCD. Wird dieser Parameter nicht angegeben, werden 10.000.000 Tests pro GVCD verwendet

-v|--verbose Wählt die Menge der während der Berechnung erzeugten Ausgaben:

0 - silent: Es wird absolut jede Ausgabe unterdrückt

1 - warning: Es werden nur Warnungen nach STDERR ausgegeben, wenn Berechnungen nicht so ablaufen wie erwartet

***2 -** normal: Es werden informative Informationen über den Programmablauf ausgegeben

3 - debug: Viele Zusatzinformationen über Zwischenergebnisse und ein Fortschrittsanzeiger innerhalb der GVCD Berechnung werden ausgegeben

«infiles» Die zu verarbeitenden Eingabedateien. Diese müssen im netCDF Format vorliegen und die in --channels angegebenen Variablen enthalten

Beschreibung: Das vorgestellte Tool implementiert alle in Kapitel 3 vorgestellten Algorithmen zur Berechnung der GVCD auf 2D, wie auch auf 3D Daten. Für eine genauere Erläuterung verweise ich daher auf die Algorithmenbeschreibungen.

getFeaVec

Kurzbeschreibung: Berechnung der Merkmalsvektoren

Syntax:

```
getFeaVec  [{-b|--greyfeanum} <number>]
           [{-c|--classwise}]
           [{-f|--feaname} <string>]
           [{-g|--grayvals} <<number>>]
           [{-l|--maskvars} <<string>>]
           [{-m|--maskfile} <string>]
           {-n|--variables} <<string>>
           [{-o|--no_constraints}]
           [{-r|--radii} <<number>>]
           [{-v|--verbose} <number>]
           [{-w|--weighing}]
           <infile>
```

Parameter:

-b|--greyfeanum Dieser Parameter wirkt nur in Verbindung mit den Parametern -m, -l und -o und wählt die Anzahl der zu berechnenden Merkmale. Es werden die angegebene Anzahl Merkmale aus der gegebenen Menge GVCDs gemäß der in -m übergebenen Maskendatei und der in -l übergebenen einzelnen Maske extrahiert. Fehlt einer der Parameter -m, -o oder -l hat -b keine Wirkung. Wird der Parameter nicht explizit angegeben liegt sein Standardwert bei 10

-c|--classwise Wird dieses Flag gesetzt, erwartet das Tool in der Maskendatei für jede Klasse eine Variable mit dem Namen `plane_<classname>`. Für jede dieser Masken wird die in -b angegebene Anzahl von Merkmalen extrahiert und zu einem Merkmalsvektor kombiniert.

-f|--feaname Der berechnete Merkmalsvektor wird in ein Attribut mit dem gegebenen Namen gespeichert. Wird der Parameter nicht explizit angegeben, wird als Standard `FeaVec` als Attributsname verwendet

-g|--grayvals Die Liste der hier übergebenen Zahlen korrespondiert zu den Grauwerten, die zur Extraktion der Merkmalsvektoren herangezogen werden. Das heißt nur die Einträge der GVCD, die zu in der Liste aufgeführten Grauwerten korrespondieren werden berücksichtigt, alle anderen übersprungen

-l|--maskvars Dieser Parameter legt die zur Merkmalsextraktion zu verwendenden Masken auf Variablenebene fest. Er wird nur in Verbindung mit dem -w Switch ausgewertet. Je nach Anzahl der übergebenen Strings verläuft die Extraktion von Merkmalen unterschiedlich:

1 Variable Gewichtung der GVCD mit anschließender Integration

n Variablen Die GVCD wird zunächst in einen Vektor umgewandelt, dann mit den einzelnen Matrizen der Masken transformiert und anschließend integriert. Dies macht Sinn, wenn die Maskendatei die Basisvektoren einer Transformation enthält und nicht direkt eine Maske, wie dies bei der KLT zum Beispiel der Fall ist

Ist eine Maskendatei gegeben, der Parameter -l bleibt aber leer, so werden alle in der Datei enthaltenen Variablen als Basisfunktionen interpretiert und entsprechend abgearbeitet

-m|--maskfile Hier übergibt man den Dateinamen der Datei, die entweder die zu verwendende Gewichtungsfunktion, oder die Basisfunktionen enthält anhand derer die Merkmale extrahiert werden (siehe -l)

-n|--variables Hier werden die netCDF Variablen-Namen übergeben, die die GVCDs enthalten, aus denen die Merkmale extrahiert werden sollen. Es ist darauf zu achten, dass die x- und y-Dimensionen bei Verwendung einer Maske mit den entsprechenden Maskendimensionen übereinstimmen müssen. Weiterhin muss die Summe der z-Dimensionen, der z-Dimension der Maske entsprechen.

-o|--no_constraints Dieser Schalter hat nur in Verbindung mit einer gegebenen Maskendatei und exakt einer Maskenvariable eine Wirkung. Sind die Voraussetzungen erfüllt, werden gemäß der Maske genau die in -b angegebene Anzahl von Features aus allen gegebenen GVCDs selektiert

-r|–radii Dieser Parameter selektiert die bei der Merkmalsberechnung verwendeten GVCD-Ebenen, also die Paramtersätze, da jede Ebene gerade durch einen Parametersatz erzeugt wird. Wird er nicht angegeben, werden alle GVCD-Ebenen zur Merkmalsextraktion verwendet

-v|--verbose Wählt die Menge der während der Berechnung erzeugten Ausgaben:

0 - silent: Es wird absolut jede Ausgabe unterdrückt

1 - warning: Es werden nur Warnungen nach STDERR ausgegeben, wenn Berechnungen nicht so ablaufen wie erwartet

***2 -** normal: Es werden informative Informationen über den Programmablauf ausgegeben

-w|--weighing Dieser Schalter wechselt vom Selektions- in den Gewichtungsmodus. Das heisst anstatt einzelne Merkmale aus den GVCDs zu selektieren werden die GVCDs gewichtet oder transformiert und anschließend ebenenweise integriert, was zu gewichteten 2-Punkt-Invarianten führt

«infiles» Die zu verarbeitenden Eingabedateien. Diese müssen im netCDF Format vorliegen und die in --variables angegebenen Variablen enthalten

Beschreibung: Die Merkmalsvektor-Berechnung ist durch Versuche mit verschiedensten Varianten der Merkmalsextraktion leider sehr unübersichtlich geworden, weswegen dieses Programm auch viele Optionen hat, die voneinander abhängen und oft bei Fehlen von zugehörigen Optionen überhaupt nichts tun, weswegen hier einige typische Anwendungsbeispiele aufgeführt sind:

Gegeben:

- Eine Datei `gvcd.nc` mit Variablen `gvcd_1` und `gvcd_2`, jeweils mit 32 GVCDs mit einer Grauwertauflösung von 16×16 Grauwerten.
- Eine Datei `weigh_mask.nc` mit Variablen `plane_1` und `plane_2` mit jeweils $16 \times 16 \times 64$ Elementen. 64 Ebenen, da Masken für beide GVCDs enthalten sind und diese einfach gestapelt werden
- Eine Datei `base.nc` mit Variablen `base_1` und `base_2` mit jeweils $16 \times 16 \times 64$ Elementen. Dabei Entspricht Ebene 1 von `base_1` der ersten Zeile der Transformationsmatrix der ersten Ebene der GVCD, `base_2` der zweiten... In diesem Fall betrachten wir also eine Basistransformation aus dem $\mathbb{R}^{256}$ in den $\mathbb{R}^2$, was bei der KLT realistisch ist. Fehlende Zeilen der Transformationsmatrix werden durch Nullen ergänzt.

Beispiel: *Selektion*
Selektion von 20 Merkmalen aus den GVCD Ebenen 1, 3 und 5 mit der Maske `plane_1` aus `weigh_mask.nc`. Der Merkmalsvektor soll nach `FeaVec_sel_20` gespeichert werden:

```
getFeaVec -f FeaVec_sel_20 -o -r 0 2 4 -b 20 \\
          -m weigh_mask.nc -l plane_1 -n gvcd_1 gvcd_2 -- gvcd.nc
```

Beispiel: *Gewichtung*
Gewichtung der GVCD Ebene 1 beider GVCDs mit den Gewichten aus `plane_2` von `weigh_mask.nc` mit anschließender Integration der einzelnen GVCDs um einen Merkmalsvektor aus 2-Punkt-Invarianten zu erhalten:

```
getFeaVec -f FeaVec_weighed -w -r 0 32 \\
          -m weigh_mask.nc -l plane_2 -n gvcd_1 gvcd_2 -- gvcd.nc
```

Beispiel: *Transformation*
Transformation der GVCDs mit Hilfe der Basisfunktionen aus `base.nc` mit anschließender Integration über die transformierten GVCDs:

```
getFeaVec -f FeaVec_transformed -w -m base.nc \\
          -l base_1 base_2 -n gvcd_1 gvcd_2 -- gvcd.nc
```

Beispiel: *Standard 2-Punkt-Invarianten*
Berechnung der Standard-2-Punkt-Invarianten mit linearer Gewichtung der Grauwerte aus den gegebenen GVCDs:

```
getFeaVec -f FeaVec_GSI -n gvcd_1 gvcd_2 -- gvcd.nc
```

Weitere Hilfsprogramme

gvcd_feaCombine

Kurzbeschreibung: Tool zum konkatenieren von netCDF-Attributen

Syntax:

```
gvcd_feaCombine  {-i|--input_features} <<string>>
                 {-o|--output_feature} <string>
                 [{-v|--verbose} <number>]
                 <infile>
```

Beschreibung: Alle in --input_features übergebenen netCDF Attribute werden aneinandergehängt und anschließend mit dem in --output_feature angegebenen Namen zurückgespeichert. Ist das Ausgabe-Attribut bereits vorhanden wird es erweitert.

selectFeatures_cmim

Kurzbeschreibung: Tool zur Merkmalsselektion mit Hilfe des cmim Algorithmus

Syntax:

```
selectFeatures_cmim  [{-c|--code} <string>]
                     [{-f|--feaname} <string>]
                     [{-l|--feature_length} <number>]
                     {-n|--variable_name} <string>
                     [{-v|--verbose} <number>]
                     <<infiles>>
```

Bemerkung: Dieses Tool ist ein Wrapper für den von François Fleuret entwickelten cmim-Feature Selektor [Fle04].

gvcd2feavec

Kurzbeschreibung: Tool zum konvertieren eines netCDF-Arrays in ein netCDF-Attribut

Syntax:

```
gvcd2feavec  [{-c|--code} <string>]
             [{-f|--feaname} <string>]
             [{-l|--feature_length} <number>]
             {-n|--variables} <<string>>
             [{-v|--verbose} <number>]
             <infiles> <outfile>
```

Beschreibung: Alle gegebenen netCDF Variablen werden Elementweise in einen Vektor geschrieben und dieser in <outfile> unter dem angegebenen Namen als Attribut abgespeichert.

getHyperplanes

Kurzbeschreibung: Tool zur Berechnung der Hyperebenen-Normalen eines gelernten SVM Modells

Syntax:

```
getHyperplanes  [{-n|--varstub} <string>]
                [{-v|--verbose} <number>]
                {-x|--xdim} <number>
                {-y|--ydim} <number>
                {-z|--zdim} <number>
                <infiles> <outfile>
```

Beschreibung: Als Eingabe nimmt das Tool ein gelerntes SVM Modell im netCDF Format, wie es das `svmtl_nc`[1] Tool ausgibt. Die Normalen werden berechnet und anschliessend in 3d-Arrays der Dimension xdim $\times$ ydim $\times$ zdim umgewandelt. Diese etwas eigentümliche Umwandlung ist praktisch, wenn man zunächst Arrays dieser Dimensionen hatte, diese mit `gvcd2feavec` in Vektoren umgewandelt hat um das SVM Modell zu lernen aber anschliessend wieder eine Voxelweise Korrespondenz zwischen Original-Arrays und Normalenelementen braucht, wie zum Beispiel zum späteren Gewichten.

generateTestPatterns

Kurzbeschreibung: Tool zum Erzeugen von Testmustern

Syntax:

```
generateTestPatterns  [{-d|--dimensions} <number>]
                      [{-n|--varname} <string>
                      [{-p|--pattern} <number>
                      [{-r|--resolution} <number>
                      [{-v|--verbose} <number>]
                      <outfile>
```

Beschreibung: Mit diesem Tool kann man verschiedene netCDF Arrays mit definierten Testdaten erzeugen. Diese beschränken sich im Moment auf sinusförmige Grauwertveränderungen, Streifenmuster und Rauschen, sind jedoch beliebig erweiterbar.

calcSqrt

Kurzbeschreibung: Berechnung der punktweisen Quadratwurzel von netCDF Arrays

Syntax:

```
calcSqrt  [{-n|--varname} <string>
          [{-v|--verbose} <number>]
          <infile>
```

Beschreibung: Dieses Tool berechnet die punktweise Quadratwurzel des in --varname übergebenen netCDF-Arrays und speichert das Ergebnis in ein Array mit Bezeichnung `sqrt_<alterName>`. Wird keine Variable angegeben, werden alle in der Datei enthaltenen Variablen verarbeitet. Da die reelle Wurzel für negative Werte nicht definiert ist, wird bei negativen Werten die Wurzel des Betrages gezogen und anschließend das alte Vorzeichen wiederhergestellt. Dies ist im mathematischen Sinn nicht korrekt, aber in diesem Kontext sinnvoll.

[1] Informationen zur libsvmtl unter http://lmb.informatik.uni-freiburg.de/lmbsoft/libsvmtl/

calcOffsLog

Kurzbeschreibung: Berechnung des punktweisen Logarithmus von netCDF Arrays

Syntax:

```
calcOffsLog  [{-a|--adjust}]
             [{-n|--variable_name} <string>
             [{-o|--offset}] <number>
             [{-v|--verbose} <number>]
             <infile>
```

Beschreibung: Dieses Tool berechnet den punktweisen Logarithmus der gegebenen netCDF-Variable. Um Probleme mit Werten < 0 zu vermeiden, wurden die Parameter -o und -a eingeführt. Der Parameter -o ermöglicht das punktweise addieren einer Konstanten vor der Logarithmierung. Der Schalter -a dient dem selben Zweck, sucht sich jedoch selbst eine geschickte Konstante um die Daten optimal wiedergeben zu können. Dies dient vorwiegend dem Zweck der Ausgabe am Bildschirm, wenn der Dynamikbereich der Variablen sehr groß ist. Mit Hilfe dieses Tools wurden alle GVCD Plots innerhalb der Arbeit vorverarbeitet.

Abbildungsverzeichnis

Literaturverzeichnis

[Bur98] Burges, Christopher J. C.: *A Tutorial on Support Vector Machines for Pattern Recognition*. Data Mining and Knowledge Discovery, 2(2):121–167, 1998.

[CTZH02] Cohen, I., Q. Tian, X. Zhou und T. Huang: *Feature selection using principal feature analysis*, 2002.

[Fle04] Fleuret, François: *Fast Binary Feature Selection with Conditional Mutual Information*. Journal of Machine Learning Research, 5:1531–1555, November 2004.

[HCS01] Heiler, Matthias, Daniel Cremers und Christoph Schnörr: *Efficient Feature Subset Selection for Support Vector Machines*. Tech. Report 21/2001, University of Mannheim, 2001. Computer Science Series.

[HSD73] Haralick, Robert M., K. Shanmugam und Its'hak Dinstein: *Textural Features for Image Classification*. IEEE Transactions on Systems, Man, and Cypernetics, 1973.

[Mit97] Mitchell, Tom M.: *Machine Learning*. McGraw-Hill, New York, 1997. Tom M. Mitchell. Includes bibliographical references and indexes. 1. Introduction – 2. Concept Learning and the General-to-Specific Ordering – 3. Decision Tree Learning – 4. Artificial Neural Networks – 5. Evaluating Hypotheses – 6. Bayesian Learning – 7. Computational Learning Theory – 8. Instance-Based Learning – 9. Genetic Algorithms – 10. Learning Sets of Rules – 11. Analytical Learning – 12. Combining Inductive and Analytical Learning – 13. Reinforcement Learning.

[PB87] Parks, T. W. und C. S. Burrus: *Digital Filter Design*. John Wiley & Sons, 1987.

[Qui86] Quinlan, J. Ross: *Induction of Decision Trees.* Machine Learning, 1(1):81–106, 1986.

[RBS02] Ronneberger, O., H. Burkhardt und E. Schultz: *General-purpose Object Recognition in 3D Volume Data Sets using Gray-Scale Invariants...* In *Proceedings of the International Conference on Pattern Recognition*, Quebec, Canada, September 2002.

[RN00] Rowland, R.E. und E.M. Nickless: *Confocal Microscopy Opens the Door to 3-Dimensional Analysis of Cells*. Bioscene, 2000.

[Ron04] Ronneberger, Olaf: *Completeness of Gray Scale Invariants*. Internal Talk, March 2004.

[Sch88] Schüssler, H. W.: *Digitale Signalverarbeitung - Band 1: Analyse diskreter Signale und Systeme*. Springer Verlag, 2 . Auflage, 1988.

[SM94] Schulz-Mirbach, H.: *Constructing Invariant Features by Averaging Techniques*. In *Proceedings of the 12th International Conference on Pattern Recognition*, Ausgabe 2, S. 387–390, Jerusalem, October 1994.

[SS99] Siggelkow, Sven und Marc Schael: *Fast estimation of invariant features*. In *Informatik aktuell*, Mustererkennung, DAGM 1999, S. 181–188. Springer, September 1999. Bonn.

[Wan05] WANG, QING: *Expression of Gray Scale Invariants with Monomial Kernels in Fourier Space*. noch nicht veröffentlicht, 2005.

[WJL97] WALKER, R., P. JACKWAY und D. LONGSTAFF: *Recent Developments In The Use Of The Co-Occurrence Matrix For Texture Recognition*. ICDSP '97, the 13th International Conference on Digital Signal Processing, Santorini, Greece, 2nd-4th June 1997.