Numerische Methoden zur Analyse hochdimensionaler Daten

Dissertation
zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
"Doctor rerum naturalium"
der Georg-August-Universität Göttingen

im Promotionsprogramm "Grundprogramm Mathematik"
der Georg August University School of Science (GAUSS)

vorgelegt von

Dennis Heinen

aus Duisburg

Göttingen, 2014

Betreuungsausschuss

- Prof. Dr. Gerlind Plonka-Hoch, Institut für Numerische und Angewandte Mathematik, Georg-August-Universität Göttingen
- Prof. Dr. Thorsten Hohage, Institut für Numerische und Angewandte Mathematik, Georg-August-Universität Göttingen

Mitglieder der Prüfungskommission

- Referentin: Prof. Dr. Gerlind Plonka-Hoch, Institut für Numerische und Angewandte Mathematik, Georg-August-Universität Göttingen
- Korreferent: Prof. Dr. Armin Iske, Fachbereich Mathematik, Universität Hamburg

Weitere Mitglieder der Prüfungskommission:

- Prof. Dr. Dorothea Bahns, Mathematisches Institut, Georg-August-Universität Göttingen
- Prof. Dr. Thorsten Hohage, Institut für Numerische und Angewandte Mathematik, Georg-August-Universität Göttingen
- Jun.-Prof. Dr. Felix Krahmer, Institut für Numerische und Angewandte Mathematik, Georg-August-Universität Göttingen
- Jun.-Prof. Dr. Andrea Krajina, Institut für Mathematische Stochastik, Georg-August-Universität Göttingen

Tag der mündlichen Prüfung: 01.07.2014

Inhaltsverzeichnis

Einleitung

In vielen modernen Anwendungsbereichen werden sehr große Datensätze produziert, welche es aufzubereiten, zu speichern und weiterzuverarbeiten gilt. Derartige Datensätze können in zweierlei Hinsicht „groß“ sein - bezüglich ihrer Kardinalität, d.h. der Anzahl anfallender Datenpunkte, und bezüglich ihrer Dimension, d.h. der Anzahl der Merkmale, welche jeder Datenpunkt aufweist. Es ist zu beachten, dass Datensätze in der Praxis im Allgemeinen nicht perfekt sind, sondern Störungen aufweisen können, welche etwa durch Übertragungsfehler oder durch Fehler und Ungenauigkeiten bei Messungen entstehen. Diese Dissertationsschrift beschäftigt sich mit zwei der wesentlichen Herausforderungen bei der Behandlung von Datensätzen, der Dimensionsreduktion und der Entstörung.

Kapitel 1 liefert eine Zusammenfassung über das Thema Dimensionsreduktion. Ziel der Dimensionsreduktion ist eine sinnvolle niedrigdimensionale Darstellung eines vorliegenden hochdimensionalen Datensatzes zur effektiveren Weiterbehandlung oder zur besseren Visualisierung der Daten. Insbesondere werden in Kapitel 1 bewährte Methoden des Manifold-Learning diskutiert und verglichen. Die zentrale Annahme des Manifold-Learning ist, dass der hochdimensionale Datensatz (approximativ) auf einer niedrigdimensionalen Mannigfaltigkeit liegt. Wir stellen hierbei die sogenannten globalen Spektralmethoden, lokalen Spektralmethoden und Nicht-Spektralmethoden gegenüber und gehen auf ihre Vor- und Nachteile ein.

In Kapitel 2 stellen wir eine neue Entstörungsmethode für hochdimensionale Daten vor (siehe auch [66]). Ziel ist die Entwicklung einer Multiskalenmethode, genauer einer Wavelet-Shrinkage-Methode, für die Glättung verrauschter Abtastwerte einer zugrundeliegenden multivariaten, reellwertigen und stückweise stetigen Funktion. Die Abtastpunkte können dabei hochdimensional und gestreut sein. Sie liegen nicht notwendig auf einem regulären Gitter. Dieses Ziel wird durch eine Verallgemeinerung und Weiterentwicklung der für die Bildkompression eingeführten „Easy Path Wavelet Transform“ (EPWT) [101] erreicht. Grundlage der Entstörungsmethode ist eine eindimensionale Wavelet-Transformation entlang (adaptiv) zu konstruierender Pfade durch die Abtastpunkte. Kapitel 2 fasst zunächst die wesentlichen Konzepte der Wavelet-Transformation, des Wavelet-Shrinkage-Ansatzes und des Cycle-Spinnings [31] zusammen. Anschließend wird das Hauptresultat der Arbeit, der Algorithmus zur Entstörung mittels Wavelets entlang von Pfaden (Algorithmus 2.63), beschrieben. Da dabei für eine erfolgreiche Datenentstörung die Wahl geeigneter Pfade, welche sich an Glattheitsstrukturen des vorliegenden Datensatzes anpassen, wesentlich ist, legen wir unser besonderes Augenmerk

auf verschiedene neue adaptive Pfadkonstruktionen. Wir diskutieren kurz theoretische Eigenschaften von Wavelets entlang von Pfaden. Weiterhin gehen wir auf Details der Implementierung des Entstörungsalgorithmus ein und präsentieren numerische Resultate, welche wir zum Nachweis der Funktionalität der Methode in Relation zu einigen vergleichbaren Entstörungsmethoden setzen. Schließlich betrachten wir einige Modifikationen des Entstörungsalgorithmus, welche sich zur weiteren Verbesserung der Entstörung oder zur Beschleunigung des Algorithmus in Betracht ziehen lassen.

Danksagungen

Diese Dissertation ist das Ergebnis einer an der Universität Duisburg-Essen, Campus Duisburg, angefangenen und am Institut für Numerische und Angewandte Mathematik der Georg-August-Universität Göttingen fortgeführten Forschungstätigkeit.

An allererster Stelle möchte ich mich herzlich bei Frau Prof. Dr. Plonka-Hoch für die exzellente Betreuung dieser Arbeit bedanken. Darüber hinaus hat sie sich über die Jahre als verständnisvolle Vorgesetzte und Kollegin erwiesen, die sich immer Zeit für Diskussionen oder zur Beseitigung von Unklarheiten genommen hat.

Auch Herrn Prof. Dr. Armin Iske, der sich als Korreferent dieser Arbeit zur Verfügung gestellt hat, und Herrn Prof. Dr. Thorsten Hohage, der als Zweitbetreuer meines Promotionsvorhabens fungiert hat, gebührt mein Dank für ihre Mühen.

Ich danke allen aktuellen und ehemaligen Kollegen in Duisburg und Göttingen, insbesondere denen der Arbeitsgruppe „Mathematical Signal and Image Processing“, für die angenehme Arbeitsatmosphäre und Zusammenarbeit.

Zu Dank verpflichtet bin ich weiter der Deutschen Forschungsgemeinschaft, welche meine Forschungstätigkeit im Rahmen des Graduiertenkollegs 1023 „Identifikation in mathematischen Modellen“ bzw. später im Rahmen des Schwerpunktprogrammes 1324 „Extraktion quantifizierbarer Information aus komplexen Systemen“ finanziell ermöglichte.

Schließlich bedanke ich mich für den außeruniversitären Rückhalt, den mir meine Freunde und meine Familie während der Erstellung dieser Arbeit gegeben haben. Vor allem bin ich meinen Eltern, welche mich auf meinem bisherigen Lebensweg in jeglicher Hinsicht bedingungslos unterstützt haben, sehr dankbar.

1 Dimensionsreduktion

Die folgende Zusammenfassung über Dimensionsreduktion und Manifold-Learning beruht hauptsächlich auf den beiden Übersichtsarbeiten von van der Maaten et al. [91] und Cayton [28] sowie dem Buch [84] von Lee und Verleysen.

1.1 Einführung

1.1.1 Dimensionsreduktion und Manifold-Learning

In vielen technischen Anwendungen der heutigen Zeit fallen große Datensätze von hoher Dimension auf (vgl. [91]), wie z.B. Sprachsignale, Datensätze in der Bildverarbeitung, bei der funktionellen Magnetresonanztomographie oder Daten, welche in virtuellen sozialen Netzwerken angesammelt werden. Hochdimensionale Datensätze sind aufgrund ihrer Größe (hohe Anzahl von Datenpunkten mit hoher Anzahl von Merkmalen) problematisch und äußerst schwierig auszuwerten. Es wirkt der sogenannte Fluch der Dimension (engl. curse of dimensionality). Der Begriff des Fluches der Dimension wurde von Bellman [15] geprägt. Die Anzahl der Abtastwerte, um einen Datensatz gut abzudecken, wächst exponentiell mit der Zahl der Dimension. Man denke an den Einheitswürfel in $\mathbb{R}^D$ versehen mit einem kartesischen Gitter mit Maschenweite $1/10$ resultierend in 10^D Gitterpunkten (vgl. [84], S.6). Weiterhin sind Intuitionen, die im 2D- oder 3D-Fall mit Hilfe einer passenden Visualisierung erhalten werden können, für höhere Dimensionen nicht mehr ohne Weiteres übertragbar (vgl. [84], S.4).

Für eine effektive Behandlung eines hochdimensionalen Datensatzes - im Rahmen von Klassifizierung, Merkmalsextraktion, Datenkompression oder Visualisierung - ist daher oft eine Dimensionsreduktion, d.h. eine sinnvolle niedrigdimensionale Darstellung der Daten, notwendig (vgl. [28]).

Ein wesentlicher Ansatz zur Dimensionsreduktion ist das sogenannte Manifold-Learning. Die meisten Datensätze hoher Dimension besitzen eine viel niedrigere intrinsische Dimension. Jeder Datenpunkt des hochdimensionalen Datensatz besteht aus unzähligen Merkmalen, von denen oft nur wenige ausreichen, um den kompletten Datensatz appro-

ximativ gut zu charakterisieren. Dies kann z.B. der Fall sein, wenn es sich bei den Merkmalen um verschiedene Messungen handelt, die auf einer gemeinsamen Ursache beruhen (vgl. [28]). Ziel ist es, die wesentlichen Informationen zu extrahieren. Manifold-Learning nimmt dazu an, dass die hochdimensionalen Daten tatsächlich oder zumindest approximativ auf einer niedrigdimensionalen Mannigfaltigkeit liegen. Wir halten zunächst die folgende Definition gemäß [28] fest.

Definition 1.1 *Es seien $d, D \in \mathbb{N}$ mit $d < D$.*

1. *Eine Abbildung $f : \mathbb{R}^D \to \mathbb{R}^d$ heißt Homöomorhpismus, wenn sie bijektiv und stetig mit ebenfalls stetiger Umkehrfunktion ist.*

2. *Eine d-dimensionale Mannigfaltigkeit $M \subset \mathbb{R}^D$ ist eine Menge, die lokal homöomorph zur offenen Einheitskugel $B_1^d(0)$ in $\mathbb{R}^d$ ist. D.h., für jedes $x \in M$ existiert eine offene Nachbarschaft $N(x)$ und ein Homöomorphismus $f_x : N(x) \mapsto B_1^d(0)$. Die Nachbarschaften $N(x)$ werden als Flicken (engl. coordinate patches), die Abbildungen f_x als Karten (engl. coordinate charts) bezeichnet. Das Bild der Karten nennt man Parameterraum.*

Beispiel 1.2 *1. Die Erdoberfläche ist modellhaft eine zweidimensionale Mannigfaltigkeit in einem dreidimensionalen Raum. Das Vorgehen, die Erdoberfläche in Form von Landkarten zweidimensional abzubilden, spiegelt sich in den Bezeichnungen aus obiger Definition wider.*

2. *Eine Schweizer Rolle (engl. Swiss roll) wie in Abbildung 1.1 ist eine einfache zweidimensionale Mannigfaltigkeit eingebettet in $\mathbb{R}^3$. Es handelt sich anschaulich, um einen Ausschnitt einer zweidimensionalen Ebene, welcher eingerollt wurde (engl. curled plane). Manifold-Learning wird gewissermaßen auf eine „Entfaltung“ bzw. ein „Entrollen“ (engl. unfolding) dieser Mannigfaltigkeit abzielen (vgl. [84], S.14 und [28, 91]).*

3. *Ein praxisnahes Beispiel findet sich [129]. Der hochdimensionale Datensatz bestehe aus 1000 Grauwert-Bildern mit je 28×28 Pixeln, welche handgeschriebene Versionen der Ziffer „2“ zeigen. Die Dimension des Datensatzes ist offenbar $28 \cdot 28 = 784$. Die Bilder unterscheiden sich im Wesentlichen jedoch nur in der Ausprägung des Bogens am oberen Ende der „2“ und der Ausprägung einer Schleife am unteren Ende der „2“. Die intrinsische Dimension des Datensatz lässt sich daher mit den Variablen „Ausprägung der unteren Schleife“ und „Ausprägung des oberen Bogens“ als 2 angeben.*

Wir formulieren nun das Problem des Manifold-Learning (vgl. [28]).

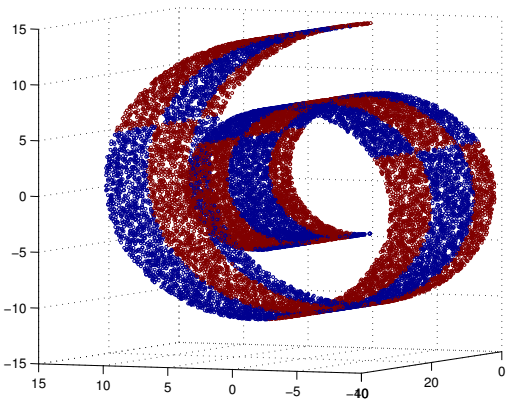

Abbildung 1.1: Mittels der Toolbox [89] erstellte Variante einer Schweizer Rolle.

Problem 1.3 *Gegeben seien Punkte $x_1, \ldots, x_n \in M \subset \mathbb{R}^D$, wobei M eine d-dimensionale Mannigfaltigkeit ($d < D$) ist, die sich mittels einer einzigen (a priori unbekannten) Karte $f : M \to \mathbb{R}^d$ beschreiben lässt. Gesucht sind nun die Punkte $y_1, \ldots, y_n \in \mathbb{R}^d$ mit*

$$y_i = f(x_i), \quad i = 1, \ldots, n.$$

Im Allgemeinen wollen bzw. müssen wir uns also damit begnügen, die niedrigdimensionalen Punkte y_i als Pendants der gegebenen hochdimensionalen Punkte x_i zu bestimmen. Wir beschränken uns daher auf eine diskrete Anschauung. Der (nicht lineare) Homöomorphismus f verbleibt im Allgemeinen unbekannt und ist durch $x_1, \ldots, x_n$ und $y_1, \ldots, y_n$ nicht eindeutig bestimmt. Die Bestimmung der niedrigdimensionalen Darstellungen zusätzlicher hochdimensionaler Punkte auf der Mannigfaltgikeit, eine sogenannte Out-of-sample-Erweiterung (engl. out-of-sample extension), ist somit in der Regel nicht trivial.

Das Problem 1.3 ist ein schlecht gestelltes inverses Problem. Wir benötigen zusätzliche Annahmen, die sicherstellen, dass die erhaltenen Punkte $y_i = f(x_i)$ eine sinnvolle niedrigdimensionale Darstellung der gegebenen hochdimensionalen Daten x_i liefern. Unterschiedliche Ansätze für diese zusätzlichen Annahmen führen zu den diversen Methoden des Manifold-Learning, von denen wir die wichtigsten hier vorstellen wollen.

Die Zieldimension (intrinsische Dimension) d ist im Allgemeinen nicht bekannt, wird jedoch als Eingabeparameter benötigt. Einige der im Folgenden beschriebenen Algorithmen des Manifold-Learning liefern als Nebenprodukt Anhaltspunkte zur Bestimmung der Zieldimension. Die Qualität dieser Anhaltspunkte leidet jedoch, falls der Datensatz mit Rauschen belegt ist. In diesem Fall muss d wie etwa in Kapitel 3 des Buches von Lee und Verleysen [84] beschrieben passend geschätzt werden. Eine einfache, jedoch auf-

grund des Aufwandes unbefriedigende, Methode ist natürlich die Schätzung von d mittels Probieren.

Zu hinterfragen ist, ob die zentrale Annahme, dass die gegebenen Daten (approximativ) auf einer Mannigfaltigkeit liegen, für natürliche Datensätze haltbar ist. Der Zusatz „approximativ“ ist besonders relevant, falls die Daten Störungen enthalten. Ob die Annahme allgemein in jedem Fall sinnvoll ist, verbleibt offen. Für einige Typen von Datensätzen wie Video- und Bild-Daten konnte sie jedoch bestätigt werden (vgl. [28]).

Für den Rest dieses Kapitels legen wir einige Notationen fest.

Notation 1.4 *1. Die gegebenen hochdimensionalen Datenpunkte seien $x_1, \ldots, x_n \in \mathbb{R}^D$ (Spaltenvektoren). Die Matrix $X = (x_1, \ldots, x_n)^T \in \mathbb{R}^{n \times D}$ bestehe aus den Zeilen $x_1^T, \ldots x_n^T$.*

2. Die niedrig-dimensionalen Zielpunkte, die Problem 1.3 (approximativ) lösen, seien $y_1, \ldots, y_n \in \mathbb{R}^d$ (Spaltenvektoren). Die Matrix $Y = (y_1, \ldots, y_n)^T \in \mathbb{R}^{n \times d}$ bestehe aus den Zeilen $y_1^T, \ldots, y_n^T$.

3. Die Nachbarschaft eines Punktes x_i im Sinne des folgenden Unterabschnittes 1.1.3 sei durch $N(x_i)$ bezeichnet.

Bevor wir uns der Vorstellung der wichtigsten Methoden des Manifold-Learning widmen, gehen wir noch auf zwei wichtige Aspekte, Vektorquantisierung und Nachbarschaften von Datenpunkten, ein, welche bei einigen Methoden später eine Rolle spielen werden.

1.1.2 Vektorquantisierung

Ist die Anzahl n der Datenpunkte sehr hoch, so kann eine Vektorquantisierung des Datensatzes nützlich sein. Vektorquantisierung (engl. vector quantization) (siehe [84], Anhang D) stellt ein eigenes Forschungsfeld dar und findet z.B. auch für Zwecke des Clusterings oder der verlustbehafteten Datenkompression Verwendung. Ziel der Vektorquantisierung ist eine Verringerung der Anzahl der Beobachtungen, d.h. der Anzahl der Datenpunkte x_i. Hierzu ersetzt man die ursprünglich gegebenen Datenpunkte $x_1, \ldots, x_n$ durch zu bestimmende sogenannte Prototypen $c_1, \ldots, c_m$ für ein zuvor festgelegtes $m < n$. Vektorquantisierung reduziert die Größe eines Datensatzes damit über eine Verringerung der Anzahl der Datenpunkte statt über eine Verringerung der Dimension der Datenpunkte und verhält sich folglich gewissermaßen komplementär zur Dimensionsreduktion (siehe [84], S.263).

Wir definieren

$$\mathrm{cod}(x_i) := \underset{1 \le j \le m}{\operatorname{argmin}} \|x_i - c_j\|.$$

Alle Punkte x_i mit $\mathrm{cod}(x_i) = j_0$ werden dann als Resultat der Vektorquantisierung ersetzt bzw. repräsentiert durch den nächstgelegenen Prototypen c_{j_0}. Man beachte, dass jeder Prototyp $c_1, \ldots, c_m$ unterschiedlich viele der Punkte $x_1, \ldots, x_n$ repräsentieren kann. Fasst man alle Punkte x_i, welche durch c_j repräsentiert werden, in einer Menge $V_j = \{x_i : \mathrm{cod}(x_i) = j\}$ zusammen, so bilden diese sogenannten Voronoi-Regionen $V_1, \ldots, V_m$ eine Zerlegung des D-dimensionalen Datensatzes (siehe [84], S.265), d.h.

$$V_{j_1} \cap V_{j_2} = \emptyset$$

für $j_1, j_2 = 1, \ldots, m,\ j_1 \neq j_2$ und

$$\bigcup_{j=1,\ldots,m} V_j = \{x_1, \ldots, x_n\}.$$

Für eine möglichst repräsentative Festlegung der Prototypen minimiert man die Verzerrung (engl. distortion) der Vektorquantisierung

$$\frac{1}{n} \sum_{i=1}^{n} \|x_i - c_{\mathrm{cod}(x_i)}\|_2^2.$$

Die Verzerrung der Vektorquantisierung mittelt also die Abstände der ursprünglichen Datenpunkte zum jeweils nächsten Prototypen, d.h. zum jeweiligen Repräsentanten (vgl. [84], S.263).

Klassische Techniken zur Minimierung der Verzerrung der Quantisierung umfassen den LBG-Algorithmus (Linde-Buzo-Gray-Algorithmus) [85], ISODATA [9] und K-means [50, 92]. Darüber hinaus existieren Techniken, die dem kompetitiven Lernen (engl. competitive learning) zuzuordnen sind. Diese nutzen ein stochastisches Gradientenverfahren (siehe [84], Anhang D).

1.1.3 Nachbarschaften

Wir unterscheiden hier drei Definitionen einer Nachbarschaft $N(x_i)$ eines Punktes $x_i \in \mathbb{R}^D$: k-Nachbarschaft, ε-Nachbarschaft und τ-Nachbarschaft, benannt nach dem jeweils charakterischen Parameter $k \in \mathbb{N}$ bzw. $\varepsilon \in \mathbb{R}_{\geq 0}$ bzw. $\tau \in \mathbb{R}_{\geq 1}$ (vgl. [84], Anhang E). Weitere ähnliche Nachbarschaftsdefinitionen sind denkbar.

k-Nachbarschaft

Die einfachste Variante ist die k-Nachbarschaft. Hier enthält $N(x_i)$ die k Punkte $x_j,\ j \in \{1,\dots,n\} \setminus \{i\}$, welche den geringsten Abstand zu x_i haben. Dabei betrachtet man üblicherweise die euklidischen Abstände.

Definition 1.5 *Für einen festen Punkt x_i und $k \in \mathbb{N}$ nennen wir*

$$N(x_i) = \{x_{i_1},\dots,x_{i_k}\} \subset (\{x_1,\dots,x_n\} \setminus \{x_i\})$$

eine k-Nachbarschaft von x_i, wenn

$$\|x_i - x_{i_l}\|_2 \leq \|x_i - x_j\|_2$$

für alle $l = 1,\dots,k$ und für alle $j \in \{1,\dots,n\} \setminus \{i, i_1,\dots,i_k\}$ gilt.

Die k-Nachbarschaft $N(x_i)$ des Punktes x_i ist folglich nicht notwendig eindeutig bestimmt, wenn mehrere Punkte gleich weit von x_i entfernt liegen. In diesem Falle wählen wir beliebig unter den möglichen k-Nachbarschaften.

Jeder Punkt hat nach dieser Nachbarschaftsdefinition genau k Nachbarn. Es gibt jedoch keine obere Schranke für die Distanz zweier Nachbarn. Weiterhin ist zu beachten, dass diese Nachbarschaftsrelation nicht symmetrisch ist. Wenn x_j in der k-Nachbarschaft von x_i ist, ist x_i nicht notwendig in der k-Nachbarschaft von x_j. Dieses Phänomen kann insbesondere auftreten, wenn x_j beispielsweise ein Ausreißer im Datensatz ist (vgl. [84], S.270 f.).

Die in vielen hier vorgestellten Dimensionsreduktionsmethoden verwendeten Nachbarschaftsgraphen, in denen zwei Punkte mit einer Kante verbunden werden, wenn sie Nachbarn sind, sind jedoch auch bei Verwendung der k-Nachbarschaft nicht gerichtet. Es findet dann eine Symmetrisierung der Nachbarschaftsrelationen statt, indem zwei Punkte x_i und x_j mit einer Kante verbunden werden, wenn x_i in der k-Nachbarschaft von x_j *oder* x_j in der k-Nachbarschaft von x_i liegt. Ein Punkt kann folglich im Nachbarschaftsgraphen mit mehr als k anderen Punkten eine Kante teilen.

ε-Nachbarschaft

Die Nachbarn eines Punktes x_i gemäß der ε-Definition einer Nachbarschaft sind die Punkte innerhalb der abgeschlossenen ε-Kugel um x_i.

Definition 1.6 *Es sei x_i ein fester Punkt und $\varepsilon > 0$. Dann heißt*

$$N(x_i) = \{x_j : \|x_i - x_j\|_2 \leq \varepsilon,\ j \neq i\}$$

die ε-Nachbarschaft von x_i.

In diesem Fall ist die Nachbarschaftsrelation symmetrisch und es existiert eine obere Schranke für die Distanz zweier Nachbarn. Allerdings kann die Kardinalität der Nachbarschaften variieren und Nachbarschaften können leer sein. Die Wahl eines passenden Parameters ist daher für die ε-Nachbarschaft schwieriger. Für die Approximation geodätischer Distanzen innerhalb einer Mannigfaltigkeit mit Hilfe eines Nachbarschaftsgraphen erzielt man mit der ε-Nachabrschaft jedoch bessere Ergebnisse als mit der k-Nachbarschaft (siehe [84], S.271).

τ-Nachbarschaft

Eine Weiterentwicklung der ε-Nachbarschaft ist in Form der τ-Nachbarschaft gegeben (siehe [84], S.271 f.).

Definition 1.7 *Für einen Datenpunkt x_i sei*

$$d_i := \min_{l=1,\ldots,n} \|x_i - x_l\|_2.$$

Zwei Datenpunkte x_i und x_j sind dann Nachbarn gemäß der τ-Nachbarschaft, wenn die Ähnlichkeitsbedingung

$$d_i \leq \tau d_j \ \wedge\ d_j \leq \tau d_i$$

und die Nachbarschaftsbedingung

$$\|x_i - x_j\| \leq \tau d_i \ \vee\ \|x_i - x_j\| \leq \tau d_j$$

erfüllt sind.

Die τ-Nachbarschaft ist ähnlich einer ε-Nachbarschaft. Der Radius der entsprechenden Kugeln ist jedoch adaptiert an die lokale Verteilung des Datensatzes, kleiner in dicht besiedelten Teilen und größer in dünn besiedelten Teilen. Die τ-Nachbarschaft ist somit aussagekräftiger betreffend der lokalen Struktur der zugrundeliegenden Mannigfaltigkeit, aber auch aufwändiger zu implementieren (vgl. [84], S.271 f.).

1.2 Dimensionsreduktionsmethoden

Es existiert eine Vielzahl von Dimensionsreduktionsmethoden und es gibt unterschiedliche Möglichkeiten, sie zu klassifizieren. In dieser Arbeit unterscheiden wir Spektralmethoden (siehe Unterabschnitte 1.2.1 und 1.2.2) und Nicht-Spektralmethoden (siehe Unterabschnitte 1.2.3 und 1.2.4). In Unterabschnitt 1.2.5 diskutieren wir noch Methoden, welche sich nicht klar in eine der genannten Kategorien einordnen lassen. Selbstverständlich ist die Auswahl der hier vorgestellten und erwähnten Dimensionsreduktionsmethoden nicht erschöpfend. Es wurden unzählige Varianten dieser Methoden sowie weitere Ansätze vorgeschlagen. Wir beschränken uns auf die grundlegendsten Methoden.

Die Spektralmethoden lösen spezifisch gestellte Optimierungsprobleme mittels Spektralzerlegung (bzw. Singulärwertzerlegung) einer entsprechenden Matrix. Globale Spektralmethoden (Unterabschnitt 1.2.1) berücksichtigen Beziehungen (wie Abstände oder Winkel) zwischen allen Datenpunkten zur Auffindung der niedrigdimensionalen Punkte y_i. Im Gegensatz dazu nutzen lokale Spektralmethoden (Unterabschnitt 1.2.2) nur die Beziehungen zwischen Datenpunkten innerhalb lokaler Nachbarschaften. Dabei entspricht die Unterscheidung zwischen „global“ und „lokal“ auch der Unterscheidung, ob eine Spektralzerlegung einer voll- oder einer dünnbesetzen Matrix - mit entsprechenden Auswirkungen auf den numerischen Aufwand - erfolgt (siehe [91]). Nicht-Spektralmethoden nutzen andere Optimierungstechniken. Hier differenzieren wir zwischen abstandserhaltenden (Unterabschnitt 1.2.3) und Topologie erhaltenden (Unterabschnitt 1.2.4) Nicht-Spektralmethoden.

1.2.1 Globale Spektralmethoden

Hauptkomponentenanalyse (PCA)

Die Hauptkomponentenanalyse (engl. Principal Component Analysis, PCA) ist die einfachste und am weitesten verbreitete Methode zum Manifold-Learning. Die PCA wurde 1901 von Pearson [98] eingeführt und von Hotelling [72] weiterentwickelt. Unabhängig davon wurde das Verfahren von Karhunen [74] beschrieben und von Loève [86] verallgemeinert. Daher ist die PCA auch unter den Namen Hotelling-Transformation oder (diskrete) Karhunen-Loève-Transformation bekannt (vgl. [84], S.24).

Die PCA ist eine lineare Methode mit der zugrundeliegenden Annahme, dass die hochdimensionalen Daten approximativ in einem linearen Unterraum niedrigerer Dimension, d.h. einer linearen Mannigfaltigkeit, liegen. Dieser lineare Unterraum wird von den Richtungen, entlang derer die Daten maximale Varianz aufzeigen, aufgespannt.

Als einfaches Beispiel (vgl. [28]) stelle man sich vor, dass die Punkte x_i in einer Ebene, d.h. einer zweidimensionalen Mannigfaltigkeit, in $\mathbb{R}^3$ liegen. In diesem Fall liefert die PCA zwei orthogonale Vektoren $v_1, v_2 \in \mathbb{R}^3$, die die Ebene aufspannen, sowie einen dritten Vektor $v_3 \in \mathbb{R}$, der orthogonal zu der Ebene steht. Offensichtlich genügen v_1 und v_2 zur Beschreibung der Mannigfaltigkeit.

Die folgende Herleitung der PCA findet sich vergleichbar in [91]. Formal beschrieben sucht die PCA die Lösung des folgenden Problems.

Problem 1.8 *Finde bei gegebener Matrix $X = (x_1, \ldots, x_n)^T \in \mathbb{R}^{n \times D}$ die lineare Abbildung $M = (m_1, \ldots, m_d) \in \mathbb{R}^{D \times d}$ mit*

$$\begin{aligned} &\operatorname{trace}(\operatorname{cov}(XM)) \to \max \\ &\|m_j\|_2^2 = 1 \quad \forall j = 1, \ldots, d. \end{aligned}$$

Dabei ist $\operatorname{cov}(XM)$ die Kovarianzmatrix von XM. Für einen d-dimensionalen Zufallsvektor $z = (z^1, \ldots, z^d)$ sind die Einträge der Kovarianzmatrix $\operatorname{cov}(z)$ definitionsgemäß die Kovarianzen der Komponenten von z, d.h.

$$\operatorname{cov}(z) = (\operatorname{cov}(z^i, z^j))_{i,j=1}^d.$$

Die Matrix $XM \in \mathbb{R}^{n \times d}$ lässt sich interpretieren als Menge von n Realisierungen eines d-dimensionalen diskreten Zufallsvektors. In diesem Sinne ist die Schreibweise $\operatorname{cov}(XM)$ zu verstehen. Die Spur $\operatorname{trace}(\operatorname{cov}(XM))$ der Kovarianzmatrix summiert dann die Varianzen der d Merkmale der transformierten Daten XM auf.

Man kann zeigen, dass sich Problem 1.8 durch eine Spektralzerlegung lösen lässt.

Satz 1.9 *Die Lösung des Optimierungsproblems 1.8 ist durch die $D \times d$-Matrix M, deren Spalten $m_1, \ldots, m_d$ die Eigenvektoren zu den d größten Eigenwerten $\lambda_1, \ldots, \lambda_d$ von $\operatorname{cov}(X)$ darstellen, gegeben.*

Beweis. Allgemein gilt

$$\operatorname{cov}(XM) = M^T \operatorname{cov}(X) M$$

für $X \in R^{n \times D}$ und $M \in \mathbb{R}^{D \times d}$. Die Hauptdiagonale von $\operatorname{cov}(XM)$ ist dann gegeben durch

$$(m_j^T \operatorname{cov}(X) m_j)_{j=1}^d.$$

Damit ist Problem 1.8 äquivalent zu

$$\sum_{j=1}^{d} m_j^T \operatorname{cov}(X) m_j \to \max \tag{1.1}$$

mit der Nebenbedingung

$$m_j^T m_j = 1 \quad \forall j = 1, \ldots, d.$$

Dieses Problem ist unter Zuhilfenahme Lagrangescher Multiplikatoren $\lambda_1, \ldots, \lambda_d$ äquivalent zum unrestringierten Optimierungsproblem

$$\sum_{j=1}^{d} (m_j^T \operatorname{cov}(X) m_j + \lambda_j (1 - m_j^T m_j)) \to \max. \tag{1.2}$$

Ableitung von (1.2) bezüglich m_j für festgelegtes $j = 1, \ldots, d$ ergibt

$$2 \operatorname{cov}(X) m_j - 2\lambda_j m_j.$$

Stationäre Punkte ergeben sich daher als Lösungen des Eigenwertproblems

$$\operatorname{cov}(X) m_j = \lambda_j m_j$$

für $j = 1 \ldots, d$. Sind $m_1, \ldots, m_D$ Eigenvektoren zu den Eigenwerten $\lambda_1 \geq \ldots \geq \lambda_D$ der Matrix $\operatorname{cov}(X) \in \mathbb{R}^{D \times D}$, so erhält man wegen

$$m_j^T \operatorname{cov}(X) m_j = \lambda_j m_j^T m_j$$

das globale Maximum des restringierten Problems (1.1), wenn man die Spalten von $M \in \mathbb{R}^{D \times d}$ als Eigenvektoren zu den d größten Eigenwerten von $\operatorname{cov}(X)$ wählt. ■

Bemerkung 1.10 *Die Spalten $m_1, \ldots, m_D$ der Matrix M sind orthonormal. Die Orthogonalität ergibt sich, da m_j Eigenvektoren der symmetrischen Matrix $\operatorname{cov}(X) \in \mathbb{R}^{D \times D}$ sind. Die Normalisierung ergibt sich durch die Nebenbedingung des restringierten Optimierungsproblems 1.8.*

Die PCA lässt sich noch auf andere Weise herleiten (siehe [84], S.26 ff.). Diesen Zugang wählte Pearson [98] ursprünglich. Man betrachte eine Kodierungsabbildung cod und eine Dekodierungsabbildung dec gemäß

$$\begin{aligned} \operatorname{cod} &: \mathbb{R}^D \to \mathbb{R}^d, \quad x \mapsto y = \operatorname{cod}(x) = M^\dagger x, \\ \operatorname{dec} &: \mathbb{R}^d \to \mathbb{R}^D, \quad y \mapsto x = \operatorname{dec}(y) = My \end{aligned}$$

mit der Pseudo-Inversen $M^\dagger = (M^T M)^{-1} M^T$ der nicht quadratischen Matrix $M \in \mathbb{R}^{D \times d}$.

Bemerkung 1.11 *1. Nach Konstruktion der Pseudo-Inversen gilt*

$$M^\dagger M = (M^T M)^{-1} (M^T M) = I_{d \times d},$$

wobei $I_{d \times d} \in \mathbb{R}^{d \times d}$ die Einheitsmatrix bezeichnet. Andersherum ist jedoch im Allgemeinen $M M^\dagger \neq I_{D \times D}$.

2. Sind die Spalten von M normiert und paarweise orthogonal, so ist

$$M^\dagger = M^T.$$

Wir formulieren nun folgendes Problem 1.12 und zeigen die Äquivalenz zu Problem 1.8.

Problem 1.12 *Gesucht ist die lineare Abbildung $M \in \mathbb{R}^{D\times d}$ mit orthonormalen Spalten, welche den über $x \in \{x_1, \ldots, x_N\}$ mittleren quadratischen Fehler*

$$E_{codec} = E[\|x - \mathrm{dec}(\mathrm{cod}(x))\|_2^2]$$

minimiert. Dabei bezeichnet $E[\cdot]$ den Erwartungswert-Operator.

Satz 1.13 *Ist der Erwartungsvektor $E[X]$ des Datensatzes $X = (x_1, \ldots, x_n)^T$ der Nullvektor, d.h.*

$$E[X] = \frac{1}{n}\sum_{i=1}^{n} x_i = 0 \in \mathbb{R}^D,$$

so sind die Probleme 1.12 und 1.8 äquivalent.

Beweis. Es gilt

$$\begin{aligned} E_{codec} =& E[\|x - MM^T x\|_2^2] \\ =& E[(x - MM^T x)^T (x - MM^T x)] \\ =& E[x^T x - 2x^T MM^T x + x^T M \underbrace{M^T M}_{=I_{d\times d}} M^T x] \\ =& E[x^T x] - E[x^T MM^T x]. \end{aligned}$$

Da $E(x^T x)$ konstant ist, ist die Minimierung von E_{codec} folglich äquivalent mit der Maximierung von

$$E[x^T MM^T x] = \frac{1}{n}\sum_{i=1}^{n} x_i^T MM^T x_i = \frac{1}{n}\sum_{i=1}^{n} (XM)_i (M^T X^T)_i = \frac{1}{n}\,\mathrm{trace}(XMM^T X^T).$$

Für die Kovarianzmatrix eines d-dimensionalen Zufallsvektors z gilt allgemein

$$\mathrm{cov}(z) = E[zz^T] - E[z](E[z])^T,$$

wobei die Erwartungswerte $E[zz^T] \in \mathbb{R}^{d\times d}$ bzw. $E[z] \in \mathbb{R}^d$ in diesem Zusammenhang komponentenweise zu verstehen sind. Wegen $E[X] = 0$ ist die Kovarianzmatrix $\mathrm{cov}(X)$ folglich gegeben durch

$$\mathrm{cov}(X) = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T = \frac{1}{n} X^T X,$$

d.h.

$$\text{cov}(XM) = \frac{1}{n}(XM)^T(XM) = \frac{1}{n}M^T X^T XM.$$

Dann ist die Maximierung von

$$\frac{1}{n}\,\text{trace}(XMM^T X^T),$$

und damit die Minimierung von E_{codec}, offensichtlich gleichbedeutend mit dem Optimierungsproblem 1.8, da $\text{trace}(AB) = \text{trace}(BA)$ allgemein für Matrizen $A \in R^{d\times n}$ und $B \in \mathbb{R}^{n\times d}$ gilt. ■

Statt der Spektralzerlegung von $\frac{1}{n}X^T X$ (d.h. von $\text{cov}(X)$, wenn $E[X] = 0$) lässt sich für die PCA alternativ die Singulärwertzerlegung von X verwenden. Die Berechnung der niedrigdimensionalen Darstellung $Y = (y_1, \ldots, y_n) \in \mathbb{R}^{n\times d}$ gemäß der PCA ist dann im folgenden Algorithmus zusammengefasst (siehe [84], S.31 f.).

Algorithmus 1.14 *1. Sei o.B.d.A. der Erwartungsvektor $E[X] = \frac{1}{n}\sum_{i=1}^{n} x_i = 0 \in \mathbb{R}^D$. Ansonsten subtrahiere $E[X]$ von jeder Zeile x_i von X.*

2. Führe eine Singulärwertzerlegung $X = W\Sigma V^T$ von X mit orthogonalen Matrizen $W \in \mathbb{R}^{n\times n}$ und $V \in \mathbb{R}^{D\times D}$ und einer Diagonalmatrix $\Sigma \in \mathbb{R}^{n\times D}$, welche die Singulärwerte von X in absteigender Reihenfolge als Einträge besitzt, durch.

3. Restringiere die Matrix V durch Streichung der $n - d$ untersten Zeilen: $VI_{D\times d}$

4. Berechne Y durch Transformation von X: $Y = XVI_{D\times d}$.

Bemerkung 1.15 *1. In Algorithmus 1.14 nutzt man eine Singulärwertzerlegung von X anstelle der von uns anfangs hergeleiteten Eigenwertzerlegung von $\text{cov}(X)$, da erstere numerisch robuster ist. Andererseits ist die Singulärwertzerlegung insbesondere für eine große Matrix X numerisch aufwändiger (siehe [84], S.32). In diesem Fall kann der Weg über die Eigenwertzerlegung von $\text{cov}(X)$ günstiger sein. Ist die Anzahl der Datenpunkte n kleiner als die Ausgangdimension D, so kann es ratsam sein, die Eigenwerte und Eigenvektoren von XX^T statt derer von $\text{cov}(X) = \frac{1}{n}X^T X$ zu berechnen. Hierzu sei auf die Ausführungen zur nächsten beschriebenen Methode, der Multidimensionalen Skalierung, verwiesen.*

2. Man beachte, dass V eine orthogonale Matrix ist. Die zugehörige Transformation stellt folglich eine Drehung der Koordinatenachsen dar.

3. Die Verteilung der Singulärwerte gibt einen Anhaltspunkt für die Zieldimension d. Liegen $x_1, \ldots, x_n$ perfekt in einem d-dimensionalen linearen Unterraum, so sind

die d größten Singulärwerte ungleich Null und die restlichen gleich Null. Außerhalb dieses perfekten Szenarios ist oft zumindest eine deutliche Lücke zwischen den d größten Singulärwerten und den restlichen sichtbar (vgl. [84], S.30).

4. *Die arithmetische Komplexität der PCA wird durch die Singulärwertzerlegung der $n \times D$-Matrix X dominiert und ist somit von der Größenordnung $\mathcal{O}(D^2)$. Der Speicherplatzbedarf liegt in der Größenordnung $\mathcal{O}(D^2)$ (siehe [91]).*

5. *Ein weiterer interessanter Aspekt bei der Dimensionsreduktion ist die Möglichkeit einer sogenannten Out-of-sample-Erweiterung (siehe [91]). Ziel hierbei ist eine Verallgemeinerung der Einbettung in Form einer nachträglichen Integration von Punkten der hochdimensionalen Mannigfaltigkeit in die gefundene niedrigdimensionale Darstellung. Für die PCA stellt dies kein Problem dar, sofern die Datenpunkte fehlerfrei in der Mannigfaltigkeit liegen. Man muss lediglich die bestimmte Abbildung auf die zusätzlichen Datenpunkte anwenden. Dieses direkte Vorgehen ist für die meisten anderen Dimensionsreduktionsmethoden nicht möglich.*

Der größte Nachteil der PCA ist ihre Linearität. Die Methode ist nicht in der Lage, die Struktur einer nicht linearen Mannigfaltigkeit vollständig zu erkennen. Für Datensätze sehr hoher Dimension (etwa $D > 50$) ist jedoch eine bewährte Methode, die PCA als Vorbehandlung zur sogenannten „harten“ Dimensionsreduktion einzusetzen, bevor man eine der hier später vorgestellten nicht linearen Dimensionsreduktionsmethoden anwendet (siehe [84], S.52 f.).

Abbildung 1.3 zeigt exemplarisch eine mittels PCA erhaltene niedrigdimensionale Darstellung der Schweizer Rolle aus Abbildung 1.2. Es verbleiben Überlagerungen, da die PCA die Rollenstruktur nicht komplett „entfalten“ kann.

Bemerkung 1.16 *Die PCA ist gewissermaßen verwandt mit der vornehmlich zur Blind-Source-Separation eingesetzten Analyse unabhängiger Komponenten (engl. Independant Component Analysis, ICA) [14]. Für die hierbei extrahierten Komponenten wird jedoch statistische Unabhängigkeit statt Orthogonalität angenommen. Die ICA lässt sich demnach auch als Weiterentwicklung bzw. Verallgemeinerung der PCA auffassen. Weiterhin besteht eine Nähe der PCA zur sogenannten Faktor-Analyse (engl. Factor Analysis [123]).*

Multidimensionale Skalierung (MDS)

Die Multidimensionale Skalierung (engl. Multidimensional Scaling, MDS) geht auf die Arbeiten von Young und Householder [143] bzw. Torgerson [133] zurück. Die MDS ver-

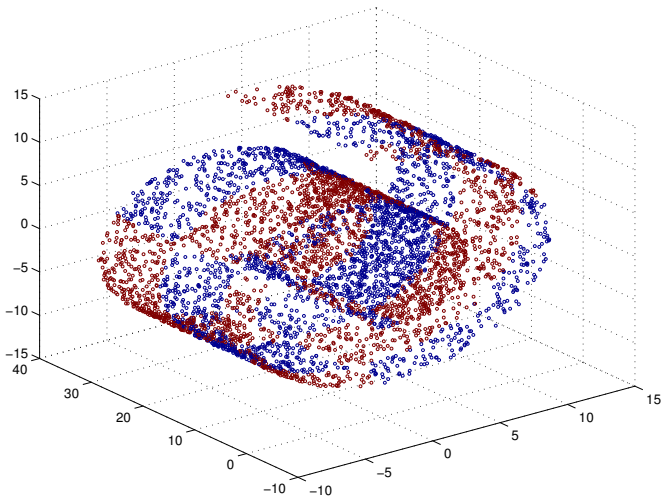

Abbildung 1.2: Mittels der Toolbox [89] erstellte Variante einer Schweizer Rolle mit 5000 verrauschten Abtastpunkten.

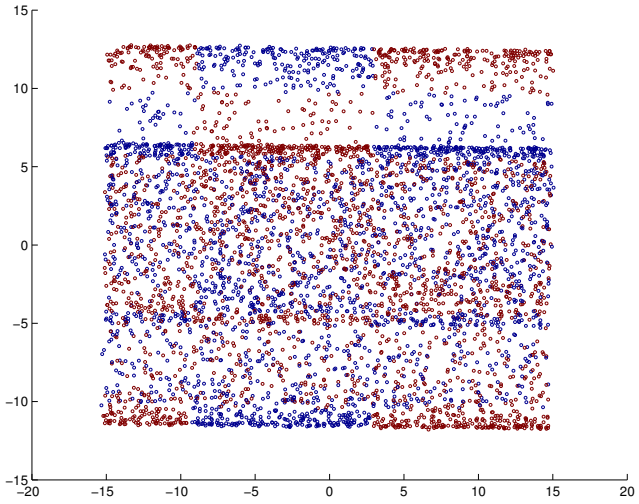

Abbildung 1.3: Mittels der Toolbox [89] erhaltene zweidimensionale Darstellung der Schweizer Rolle gemäß PCA.

sucht die paarweisen Abstände der gegebenen hochdimensionalen Punkte in der niedrigdimensionalen Einbettung beizubehalten. Man sucht eine Einbettung derart, dass

$$\phi(Y) = \sum_{i,j} \left(\|x_i - x_j\|_D - \|y_i - y_j\|_d\right)^2$$

minimal wird.

Bei der klassischen metrischen MDS wählt man für beide Normen $\|\cdot\|_d$ und $\|\cdot\|_D$ den euklidischen Abstand in $\mathbb{R}^d$ bzw. $\mathbb{R}^D$. Für die Norm $\|\cdot\|_D$ sind prinzipiell unterschiedliche Wahlen denkbar. Hier sei z.B. auf die später vorgestellte Methode Isomap verwiesen.

Das Minimierungsproblem lässt sich mittels Spektralzerlegung der Matrix der paarweisen Abstände $\mathfrak{D} = (\|x_i - x_j\|_D)_{i,j=1}^n \in \mathbb{R}^{n\times n}$ lösen. Theoretische Grundlage dafür ist der an die nachfolgende Definition anschließende Satz (siehe [28]).

Definition 1.17 *1. Es sei $H = I - \frac{1}{n}\mathbb{1}\mathbb{1}^T \in \mathbb{R}^{n\times n}$ mit $\mathbb{1} = (1,\ldots,1)^T \in \mathbb{R}^n$ die sogenannte Zentrierungsmatrix.*

2. Eine Matrix $\mathfrak{D} \in \mathbb{R}^{n\times n}$ heißt euklidische Distanzmatrix, falls Punkte $x_1,\ldots,x_n \in \mathbb{R}^D$ derart existieren, dass $\mathfrak{D} = (\|x_i - x_j\|_2)_{i,j=1}^n$ gilt.

Satz 1.18 *Eine nicht negative, symmetrische Matrix $\mathfrak{D} \in \mathbb{R}^{n\times n}$ mit Nullen auf der Hauptdiagonalen ist eine euklidische Distanzmatrix genau dann, wenn $B := -\frac{1}{2}H\mathfrak{D}H$ positiv semi-definit ist. In diesem Fall ist B die Gramsche Matrix einer um ihren Erwartungsvektor zentrierten Konfiguration von Punkten, innerhalb derer die gegenseitigen Abstände durch die Matrix $\mathfrak{D}$ gegeben sind.*

Die Matrix B lässt sich folglich als die Gramsche Matrix unserer gesuchten Punktmenge $y_1,\ldots,y_n$ auffassen, d.h. $B = YY^T$. Um nun Y zu bestimmen, gehen wir wie im folgenden Algorithmus 1.19 beschrieben vor (siehe [28]).

Algorithmus 1.19 *1. Setze $B := -\frac{1}{2}H\mathfrak{D}H$.*

2. Berechne die Spektralzerlegung $B = U\Lambda U^T$ der symmetrischen Matrix B mit einer orthogonalen Matrix U und der Diagonalmatrix Λ, die die Eigenwerte von B in absteigender Reihenfolge als Einträge enthält.

3. Definiere Λ_+ mittels $[\Lambda_+]_{ij} = \max\{[\Lambda]_{ij}, 0\}$.

4. Berechne $U\Lambda_+^{1/2}$ und bestimme Y durch Streichung der $n-d$ untersten Zeilen: $Y = U\Lambda_+^{1/2}I_{n\times d}$.

Bemerkung 1.20 *1. Die Größenordnung der durch die Spektralzerlegung dominierten arithmetischen Komplexität des Algorithmus beträgt $\mathcal{O}(n^3)$, die Größenordnung des Speicherplatzbedarfs $\mathcal{O}(n^2)$ (siehe [91]).*

2. Schritt 3 des obigen Algorithmus ist nur für den Fall relevant, in dem die Eingabematrix $\mathfrak{D}$ etwa durch Rauschen bedingt keine euklidische Distanzmatrix ist. In diesem Fall ist $\mathfrak{D}$ nach Satz 1.18 nicht positiv semi-definit und wird mittels Schritt 3 auf den Kegel der positiv semi-definiten Matrizen projiziert (siehe [28]).

3. In der Tat ist das Ergebnis der MDS für zentrierte Datensätze mit dem Ergebnis der zuvor vorgestellten Methode der PCA identisch. Die Eigenvektoren v_i der Matrix X^TX und die Eigenvektoren u_i der Gramschen Matrix XX^T erfüllen die Beziehung

$$\sqrt{\lambda_i}u_i = Xv_i \tag{1.3}$$

(siehe [91]). Dabei haben X^TX und XX^T dieselben Eigenwerte λ_i. Nun entspricht die linke Seite von (1.3) gerade dem Resultat der MDS (siehe Algorithmus 1.19, Schritt 4) und die rechte Seite dem der PCA (siehe Algorithmus 1.14, Schritt 4). Wie die PCA ist die klassische metrische MDS eine lineare Dimensionsreduktionsmethode.

4. Bezüglich einer Verallgemeinerung der gefundenen niedrigdimensionalen Darstellung auf neue Testpunkte (Out-of-sample-Erweiterung) ist bei der MDS zu unterscheiden, wie die Datenpunkte gegeben sind. Liegen sie in Form von Koordinaten vor, so lässt sich wie bei der zur MDS äquivalenten PCA ohne Weiteres die bestimmte lineare Abbildung auf die zusätzlichen Testpunkte anwenden. Sind uns hingegen nur die paarweisen Abstände oder paarweisen Skalarprodukte der Datenpunkte gegeben, so ist ein auf der Nyström-Formel [8] beruhendes Vorgehen notwendig (siehe [84], S.78 f. und [16]).

In Abbildung 1.4 findet man beispielhaft eine mit der MDS gefundene niedrigdimensionale Darstellung der Schweizer Rolle aus Abbildung 1.2. Das Ergebnis ist wie zuvor theoretisch begründet äquivalent mit dem Ergebnis der PCA (siehe Abbildung 1.3).

Bemerkung 1.21 *1. Varianten der metrischen MDS minimieren ein modifiziertes Fehlerfunktional der Form*

$$\phi(Y) = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\|x_i - x_j\|_D - \|y_i - y_j\|_d\right)^2,$$

wobei $w_{ij} \geq 0$ Gewichte sind, mittels derer gewisse Abstände stärker berücksichtigt werden als andere. Die populärste dieser Varianten ist die sogenannte Sammon-Abbildung (engl. Sammon Mapping), welche in dieser Arbeit noch diskutiert wird (vgl. [84], S.80 f.).

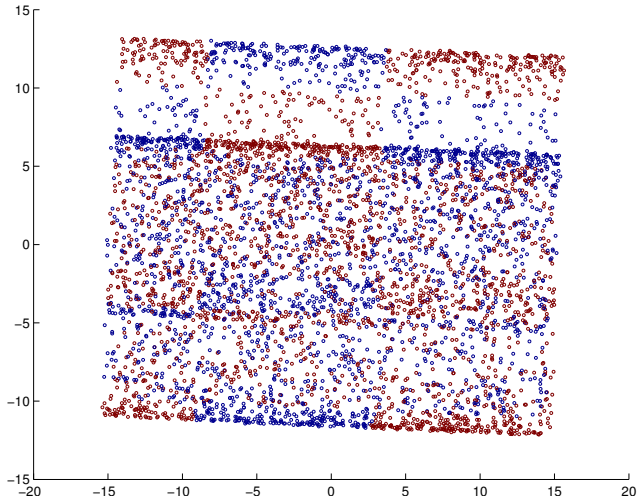

Abbildung 1.4: Mittels der Toolbox [89] erhaltene zweidimensionale Darstellung der Schweizer Rolle gemäß MDS.

2. *Eine weitere Verallgemeinerung, die sogenannte nicht metrische MDS, geht auf Shepard [121] und Kruskal [77] zurück. Hierbei sind keine Abstände der Datenpunkte $x_1, \ldots, x_n$ gegeben, sondern nur Werte $\delta(x_i, x_j)$, die die Ähnlichkeit des Paares (x_i, x_j) messen. Dieses Ähnlichkeitsmaß wird mittels einer monotonen Transformation ρ mit $\rho(\delta(x_i, x_j)) \approx \|x_i - x_j\|_D$ in Abstände übersetzt. Anschließend minimiert man ein Fehlerfunktional der Form*

$$\phi(Y) = \sqrt{\frac{\sum_{i,j=1}^{n} w_{ij} \left|\rho(\delta(x_i, x_j)) - \|y_i - y_j\|_d\right|^2}{\sum_{i,j=1}^{n} w_{ij} \|x_i - x_j\|_D}}$$

(siehe [84], S.81).

Isometrische Merkmalsabbildung (Isomap)

Der euklidische Abstand zweier hochdimensionaler Punkte $x_i, x_j \in M \subset \mathbb{R}^D$ auf einer d-dimensionalen Mannigfaltigkeit M kann unzureichend sein, um ihre Lage in Bezug auf die Mannigfaltigkeit zu charakterisieren, da er entlang der Geraden zwischen x_i und x_j im $\mathbb{R}^D$ gemessen wird. Man kann stattdessen Abstände entlang der Mannigfaltigkeit messen, die sogenannten geodätischen Distanzen. Die geodätische Distanz ist das Minimum der Integrale der Norm der Jacobi-Matrix der Parametrisierung von M über allen Pfaden in M, die x_i und x_j verbinden. Man beachte, dass die Mannigfaltigkeit, die wir rekonstruieren wollen, bekannt sein müsste, um den exakten geodätischen Abstand zweier Punkte zu berechnen. Doch selbst, wenn uns die Mannigfaltigkeit gegeben

wäre, verbleibt die Berechnung der geodätischen Distanz zweier Punkte aufwändig bzw. unmöglich (vgl. [84], S.99 ff.).

Die isometrische Merkmalsabbildung (engl. Isometric Feature Mapping, kurz Isomap) wurde von Tenenbaun et al. in [128, 129] eingeführt und stellt eine Variante der MDS dar, bei der die Norm $\|x_i - x_j\|_D$ durch eine Approximation der geodätischen Distanz von x_i und x_j ersetzt wird. Isomap besteht aus zwei Schritten.

1. Schätze die geodätischen Distanzen der hochdimensionalen Punkte x_i mittels sogenannter Graph-Distanzen.

2. Führe eine MDS mit den bestimmten Graph-Distanzen durch. Man finde also niedrigdimensionale Punkte $y_i \in \mathbb{R}^d$, deren euklidische Abstände den Graph-Distanzen der Punkte $x_i \in \mathbb{R}^D$ möglichst gut entsprechen.

Da die Berechnung der Graph-Distanzen $\mathcal{O}(n^2 \log n)$ Operationen benötigt (siehe [84], S.107), ist die arithmetische Komplexität von Isomap von der Größe $\mathcal{O}(n^3)$. Die Größenordnung des Speicherplatzbedarfs beträgt $\mathcal{O}(n^2)$ (siehe [91]).

Zur Bestimmung der Graph-Distanzen als Approximation der geodätischen Distanzen geht man wie folgt vor:

1. Erstelle einen gewichteten Graph G, dessen Ecken die Punkte $x_1, \ldots, x_n$ repräsentieren. Jeder Punkt x_i erhält Kanten zu seinen Nachbarpunkten, die gemäß Unterabschnitt 1.1.3 berechnet werden. Das Gewicht einer Kante zwischen benachbarten Knoten x_i und x_j wird jeweils durch den euklidischen Abstand $\|x_i - x_j\|_2$ festgelegt.

2. Wende den Algorithmus von Dijkstra [45] auf den Graph G an, um den kürzesten Pfad zwischen zwei Punkten x_i und x_j zu finden. Die Länge $d_G(i, j)$ dieses kürzesten Pfades, d.h. die Summe der Gewichte der Kanten entlang des Pfades im Graphen G, ist die Graph-Distanz von x_i und x_j, d.h. die Schätzung für die geodätische Distanz.

Zu beachten ist, dass Isomap trotz der Nutzung lokaler Nachbarschaften eine globale Dimensionsreduktionsmethode ist. Die Nachbarschaften werden nur zur Approximation der geodätischen Distanzen benötigt. Für die Berechnung der niedrigdimensionalen Darstellung des Datensatzes werden dann die paarweisen Abstände *aller* Punkte berücksichtigt. Die Berechnung einer niedrigdimensionalen Darstellung mittels Isomap kann recht langsam werden, wenn viele Datenpunkte vorhanden sind.

Isomap besitzt eine (asymptotische) Optimalitätsgarantie, die absichert, dass unter gewissen Bedingungen an die Mannigfaltigkeit die Parametrisierung der Mannigfaltigkeit erkannt wird (siehe [28]). Grundlage dafür ist, dass Bernstein et al. in [17] zeigen können,

dass für eine isometrisch eingebettete, kompakte Mannigfaltigkeit in $\mathbb{R}^D$ mit konvexem Parameterraum, die genügend gut abgetastet ist, die Graph-Distanzen d_G asymptotisch gegen die tatsächliche geodätische Distanz konvergieren. In der Praxis hängt es vom vorliegenden Datensatz einerseits und von der Wahl des Parameters k bzw. ε für die Nachbarschaftsdefinition andererseits ab, wie gut die Graph-Distanten die geodätischen Distanzen approximieren (vgl. [84], S.102). Umgekehrt sei erwähnt, dass das Manifold-Learning mittels Isomap fehlschlagen kann, falls die Mannigfaltigkeit nicht konvex ist (siehe [91]).

Als eine MDS unter Verwendung von Graph-Distanzen stellt Isomap eine relativ einfache Technik dar. Im Gegensatz zur klassischen metrischen MDS handelt es sich bei Isomap jedoch um eine nicht lineare Dimensionsreduktionsmethode. Es können also grundsätzlich auch nicht lineare Mannigfaltigkeiten erkannt werden. Isomap verfügt nicht über eine eingebaute Möglichkeit einer Verallgemeinerung der niedrigdimensionalen Darstellung auf zusätzliche Datenpunkte. Eine solche Out-of-sample-Erweiterung kann jedoch über die Nyström-Formel erhalten werden (siehe [84], S.108 und [16]).

Ein Nachteil von Isomap ist die topologische Instabilität. Bei der Konstruktion der Graph-Distanzen können fehlerhafte Verbindungen, sogenannte „Kurzschlüsse“ (engl. short circuits), auftreten, insbesondere wenn der Datensatz Rauschen oder Ausreißer (engl. outliers) enthält oder die Mannigfaltigkeit nicht konvex ist. Eine fehlerhafte Verbindung im Nachbarschaftsgraphen kann verheerende Auswirkungen auf die Güte der Approximation der geodätischen Distanz haben. Im Falle einer Mannigfaltigkeit mit Löchern kann man dem entgegenwirken, indem man die Mannigfaltigkeit unterteilt und sie stückweise behandelt (siehe [91]). Kritisch für die Approximation der geodätischen Distanz ist weiter, wie dicht die Mannigfaltigkeit abgetastet ist (vgl. [84], S.103). Hier wirkt auch der Fluch der Dimension (engl. curse of dimensionality). Die Anzahl der zu hinreichender Charakterisierung der Mannigfaltigkeit benötigten Abstastpunkte wächst exponentiell mit der intrinsischen Dimension. Weiterhin ist Isomap eine globale Methode des Manifold-Learning. Durch die Verwendung der MDS werden bevorzugt große paarweise Graph-Distanzen erhalten, wodurch lokale Strukturen nicht immer optimal erfasst werden können (vgl. [91]).

Abbildung 1.5 zeigt eine mittels Isomap erhaltene niedrigdimensionale Darstellung der Schweizer Rolle aus Abbildung 1.2. Für die Nachbarschaftsdefinition wurden die $k = 12$ nächsten Nachbarn betrachtet. Isomap legt die Struktur als Ebene nahezu perfekt offen. Etwas störend sind dabei die kleineren Löcher, welche die Darstellung aufweist.

Hauptkomponentenanalyse mit Kernen (KPCA)

Wie der Name suggeriert, stellt die Hauptkomponentenanalyse mit Kernen (engl. Kernel PCA, KPCA) [118] eine Verallgemeinerung der PCA dar. Sie beruht auf dem so-

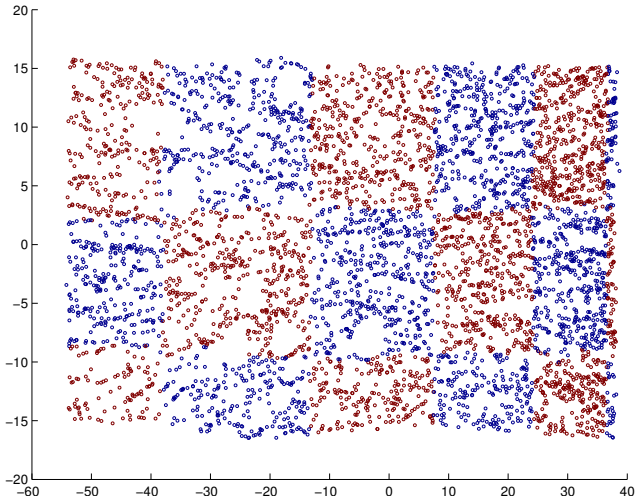

Abbildung 1.5: Mittels der Toolbox [89] erhaltene zweidimensionale Darstellung der Schweizer Rolle gemäß Isomap.

genannten „Kern-Trick“ (engl. kernel trick) und benutzt die Spektralzerlegung einer Kern-Matrix anstelle der Spektralzerlegung der Kovarianzmatrix bei der PCA. Genauer wird bei der KPCA eine PCA in einem hochdimensionalen Raum durchgeführt, der durch eine gegebene Kern-Funktion κ konstruiert wird (vgl. [91]).

Anders als den meisten übrigen vorgestellten Methoden liegt der KPCA keine direkte geometrische Motivation zugrunde. Ziel ist es, die Mannigfaltigkeit M zu linearisieren. Dabei wird die Mannigfaltigkeit mittels einer Abbildung $\phi : M \to \mathbb{R}^Q$ in einen durch einen Kern induzierten (hochdimensionalen) Merkmalsraum transformiert, in dem eine (lineare) PCA besser durchführbar ist. Dabei kann durchaus $Q > D$ sein (siehe [84], S.120). Hierdurch wird die KPCA zu einer nicht linearen Dimensionsreduktionsmethode. Eine explizite Form der Abbildung ϕ ist im Allgemeinen nicht bekannt. Wir nehmen jedoch an, dass die Skalarprodukte $\langle \phi(x_i), \phi(x_j) \rangle$ durch

$$\kappa(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

gegeben sind.

Dabei kann κ eine beliebige Kern-Funktion $\kappa : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ mit der Eigenschaft sein, dass die Kern-Matrix $K = (k_{ij})_{i,j=1}^n$ mit

$$k_{ij} = \kappa(x_i, x_j)$$

positiv semi-definit ist. Der Wert k_{ij} ist das innere Produkt von x_i mit x_j in dem durch die Kern-Funktion κ konstruierten hochdimensionalen Vektorraum. Üblich sind polynomiale Kern-Funktionen

$$\kappa(x_i, x_j) = \langle x_i, x_j \rangle^p,$$

wobei $p \in \mathbb{N}$ und $\langle \cdot, \cdot \rangle$ das Standardskalarprodukt in $\mathbb{R}^D$ ist, oder ein Gauß-Kern

$$\kappa(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}\right)$$

mit einem Parameter $\sigma > 0$. Wählt man eine lineare Kern-Funktion, d.h. eine polynomiale Kern-Funktion mit $p = 1$, so ist die Kern-Matrix die Gramsche Matrix von $\{x_1, \ldots, x_n\}$ und die KPCA ist identisch mit der (linearen) PCA (siehe [91]). Die Wahl einer passenden Kern-Funktion mitsamt ihrer Parameter für den jeweiligen Datensatz ist wesentlich für den Erfolg der KPCA (vgl. [84], S.124 f.).

Bei der KPCA wird die Matrix K zunächst zentriert gemäß

$$\tilde{k}_{ij} = -\frac{1}{2}\left(k_{ij} - \frac{1}{n}\sum_{l=1}^{n} k_{il} - \frac{1}{n}\sum_{l=1}^{n} k_{jl} + \frac{1}{n^2}\sum_{l,m=1}^{n} k_{lm}\right)$$

und es werden die größten d Eigenwerte λ_i und zugehörigen Eigenvektoren v_i der zentrierten Matrix $\tilde{K} = (\tilde{k}_{ij})_{i,j=1}^n$ errechnet. Die Eigenvektoren $a_i = (a_i^{(j)})_{j=1}^n$ der Kovarianzmatrix der Daten im durch κ konstruierten Merkmalsraum erhält man dann durch Normierung der Eigenvektoren v_i als

$$a_i = \frac{1}{\sqrt{\lambda_i}} X v_i.$$

Für die niedrigdimensionale Darstellung Y projiziert man schließlich die Ausgangsdaten auf die errechneten Eigenvektoren a_i der Kovarianzmatrix im Merkmalsraum, d.h.

$$y_i = \left(\sum_{j=1}^{n} a_1^{(j)} \kappa(x_j, x_i), \ldots, \sum_{j=1}^{n} a_d^{(j)} \kappa(x_j, x_i)\right)^T$$

(siehe [91]).

Bemerkung 1.22 *1. Die Größe der Kern-Matrix hängt nicht von der Ausgangsdimension D des Raumes, sondern von der Anzahl n der betrachteten Punkte x_i ab. Die arithmetische Komplexität der KPCA liegt in der Größenordnung $\mathcal{O}(n^3)$, der Speicherplatzbedarf ist von der Größe $\mathcal{O}(n^2)$ (siehe [91]).*

2. Es sei noch angemerkt, dass sich die bereits vorgestellte Methode Isomap sowie die in dieser Arbeit noch folgenden Methoden der lokal linearen Einbettung (LLE) und der Laplacesche Eigenabbildungen als eine KPCA mit einem speziellen Kern interpretieren lassen (siehe [62]). Man nennt nicht lineare Algorithmen zum Manifold-Learning dieser Art daher mitunter auch Kern-Methoden.

3. Eine Verallgemeinerung der durch eine KPCA gefundenen niedrigdimensionalen Darstellung auf neue Punkte (Out-of-sample-Erweiterung) lässt sich durch Modifikation des Vorgehens für die MDS erhalten (siehe [84], S.124).

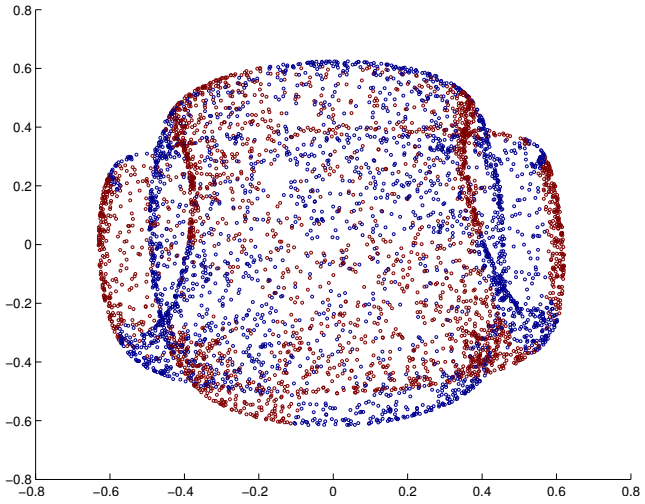

Abbildung 1.6: Mittels der Toolbox [89] erhaltene zweidimensionale Darstellung der Schweizer Rolle gemäß KPCA.

Trotz des theoretisch vielversprechenden Ansatzes hat die KPCA nur in wenigen Anwendungen zufriedenstellende Ergebnisse geliefert. Die Auswahl des passenden Kerns erweist sich als aufwändig und es fehlt die nötige geometrische Interpretation (siehe [91] und [84], S.124 f.). Lee und Verleysen (siehe [84], S.125) sehen die KPCA daher eher als einen theoretischen Rahmen und empfehlen, diese Methode in der Praxis nicht direkt zur Dimensionsreduktion zu verwenden.

Eine durch die KPCA erhaltene niedrigdimensionale Darstellung der Schweizer Rolle aus Abbildung 1.2 findet man in Abbildung 1.6. Die Darstellung weist starke Überlagerungen auf und kann daher kaum überzeugen. Als Kern-Funktion wurde der Gauß-Kern mit Parameter $\sigma = 10$ benutzt. Die Wahl des Parameters σ nimmt in diesem Beispiel sehr großen Einfluss auf die Form der Darstellung und ist daher kritisch.

Entfaltung mit maximaler Varianz (MVU)

Die Entfaltung mit maximaler Varianz (engl. Maximum Variance Unfolding, MVU) [140] ist eine Dimensionsreduktionsmethode, die versucht, die Abstände zwischen benachbarten Punkten x_i bestmöglich zu erhalten, gleichzeitig jedoch auch die restlichen paarweisen Abstände zu maximieren. Dadurch wird die Mannigfaltigkeit gewissermaßen „entfaltet“ (engl. unfolded), wobei die lokale Geometrie der Mannigfaltigkeit möglichst gut erhalten bleiben soll (vgl. [28]).

Zunächst konstruiert man bei der MVU einen Nachbarschaftsgraphen G mit den Knoten $x_1, \ldots, x_n$, wobei jeder Knoten mit seinen Nachbarn verbunden wird. Für die MVU ist nun folgendes Optimierungsproblem zu lösen.

Problem 1.23 *Finde $y_1, \ldots, y_n$, sodass*

$$\begin{aligned}
&\sum_{i,j=1}^{n} \|y_i - y_j\|_2^2 \to \max, \\
&\|y_i - y_j\|_2^2 = \|x_i - x_j\|_2^2, \ \textit{falls } (i,j) \in G, \\
&\sum_{i=1}^{n} y_i = 0.
\end{aligned}$$

Dabei bedeutet $(i,j) \in G$, dass zwischen den Knoten x_i und x_j eine Kante im Graphen G existiert. Die zweite Nebenbedingung wird benötigt, um eine Invarianz der Lösung $y_1, \ldots, y_n$ bezüglich Translation zu beseitigen (siehe [84], S.127).

Dieses Optimierungsproblem lässt sich in ein sogenanntes semi-definites Programm (engl. Semi Definite Program, SDP) übersetzen, weshalb die MVU auch unter dem Namen semi-definite Einbettung (engl. Semi Definite Embedding, SDE) bekannt ist. Für Details zu semi-definiten Programmen sei etwa auf [135] verwiesen. Mit der Matrix $K = (k_{ij})_{i,j=1}^n$ der inneren Produkte $k_{ij} = \langle y_i, y_j \rangle$ der niedrigdimensionalen Darstellung Y ist

$$\|y_i - y_j\|_2^2 = \langle y_i, y_i \rangle + \langle y_j, y_j \rangle - 2\langle y_i, y_j \rangle = k_{ii} + k_{jj} - 2k_{ij}.$$

Damit gelten die Gleichheiten

$$\begin{aligned}
&\sum_{i,j=1}^{n} \|y_i - y_j\|_2^2 = 2n \sum_{i=1}^{n} k_{ii} - 2 \sum_{i,j=1}^{n} k_{ij} = 2n \operatorname{trace}(K) - 2 \sum_{i,j=1}^{n} k_{ij} \\
&\|y_i - y_j\|^2 = k_{ii} + k_{jj} - 2k_{ij} \\
&\sum_{i,j=1}^{n} k_{ij} = \sum_{i,j=1}^{n} \langle y_i, y_j \rangle = \langle \sum_{i=1}^{n} y_i, \sum_{j=1}^{n} y_j \rangle.
\end{aligned}$$

Als Folge ist das Problem (1.23) äquivalent zu folgendem semi-definiten Programm.

Problem 1.24 *Finde* $K = (k_{ij})_{i,j=1}^{n} \in \mathbb{R}^{n \times n}$, *sodass*

$$\begin{aligned} &\operatorname{trace} K \to \max, \\ &k_{ii} + k_{jj} - 2k_{ij} = \|x_i - x_j\|^2, \text{ falls } (i,j) \in G, \\ &\sum_{i,j=1}^{n} k_{ij} = 0, \\ &K \text{ symmetrisch und positiv semi-definit.} \end{aligned}$$

Die MVU lernt also eine passende Kern-Matrix K, welche bei der zuvor diskutierten KPCA explizit vorgegeben werden muss, aus den konkret vorliegenden Daten. Aus der Lösung $K = Y^T Y$ des semi-definiten Programms lässt sich Y mittels Spektralzerlegung analog zum Vorgehen bei der MDS bestimmen.

Bemerkung 1.25

1. *Die MVU ist gemäß des zu lösenden Maximierungsproblems gewissermaßen eine Mischform aus lokaler und globaler Methode und verhält sich sogar etwas mehr wie eine lokale Methode (siehe [28]). Wir ordnen die MVU hier dennoch den globalen Methoden zu, da im letzten Schritt des Algorithmus die Spektralzerlegung einer vollbesetzten Matrix erfolgt.*

2. *Wie bei Isomap kann das Ergebnis des Manifold-Learning unter fehlerhaften Verbindungen, den „Kurzschlüssen", im Nachbarschaftsgraphen leiden, vor allem wenn im Datensatz Rauschen oder Ausreißer vorkommen. Die hierdurch entstehenden fehlerhaften Nebenbedingungen behindern die semi-definite Programmierung (siehe [91]).*

3. *Der größte Nachteil der MVU ist jedoch, dass die Methode aufgrund des Aufwands der semi-definiten Programmierung langsam bzw. für große Datensätze nicht durchführbar ist (vgl. [84], S.131). Die arithmetische Komplexität und der Speicherplatzbedarf sind von der Ordnung $\mathcal{O}(k^3 n^3)$ (siehe [91]), wobei der Parameter k die Größe der Nachbarschaften angibt. Eine effizientere Variante stellt die MVU mit sogenannten Landmarken (engl. landmarks) [139] dar. Hierbei werden nicht alle paarweisen Abstände benutzt, sondern nur Abstände zu festgelegten Landmarken. Das Problem wird auf diese Weise approximativ gelöst.*

4. *Eine Verallgemeinerung der gefundenen niedrigdimensionalen Darstellung auf neue Punkte (Out-of-sample-Erweiterung) ist im Gegensatz zu den meisten anderen Spektralmethoden nicht über die Nyström-Formel möglich, da die von der MVU gelernte Kern-Funktion im Allgemeinen nicht explizit bekannt ist (siehe [84], S.130). Hier sind andere approximative Ansätze nötig (siehe [91]).*

Abbildung 1.7 zeigt eine mittels MVU berechnete niedrigdimensionale Darstellung der Schweizer Rolle aus Abbildung 1.2. Es wurde der Nachbarschaftsparameter $k = 12$

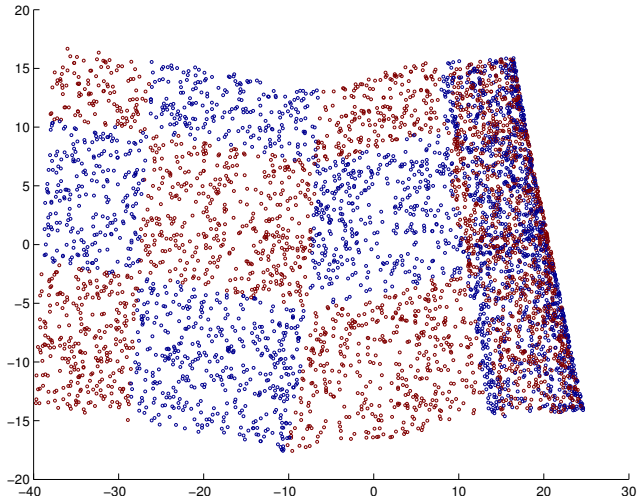

Abbildung 1.7: Mittels der Toolbox [89] erhaltene zweidimensionale Darstellung der Schweizer Rolle gemäß MVU.

gewählt. Bis auf den Streifen am rechten Ende, in den offenbar übermäßig viele Punkte abgebildet wurden, wird die intrinsische Struktur der Rolle hier gut „entfaltet“. Es sei darauf hingewiesen, dass die MVU aus numerischen Kostengründen in der verwendeten Matlab-Toolbox [89] als Erweiterung der Methode der lokal linearen Einbettung (LLE) implementiert ist.

Diffusionsabbildungen (Diffusion Maps)

Die Diffusionsabbildungen (engl. Diffusion Maps) [32, 33, 80] basieren auf der Theorie dynamischer Systeme. Diese Methode sucht eine isometrische Einbettung unter Beibehaltung bestimmter paarweiser Abstände, den sogenannten Diffusionsdistanzen (engl. diffusion distances), der Punkte. Dieser Abstandsdefinition liegen die Übergangswahrscheinlichkeiten einer Markovschen Zufallsbewegung (engl. random walk) durch die Datenpunkte zugrunde (siehe [91]).

Wir widmen uns zunächst der Definition der Diffusionsdistanz. Man erstelle zunächst einen vollständigen Graph G, dessen Knoten die Punkte $x_1, \ldots, x_n$ sind. Die Kantengewichte w_{ij} setzt man mittels eines Gauß-Kerns mit Varianz σ^2 als

$$w_{ij} = \exp\left(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}\right)$$

fest. Durch Normalisierung definieren wir

$$p_{ij}^{(1)} = \frac{w_{ij}}{\sum_{k=1}^{n} w_{ik}}.$$

Dies lässt sich interpretieren als die Übergangswahrscheinlichkeit vom Punkt x_i zum Punkt x_j in einem Zeitschritt $t = 1$ in einem dynamischen Prozess. Punkte x_i und x_j, die bezogen auf ihren euklidischen Abstand nah beieinander liegen, besitzen eine hohe Übergangswahrscheinlichkeit $p_{ij}^{(1)}$ und weiter voneinander entfernte Punkte eine geringere (siehe [91]).

Die zugehörige Matrix $P^{(1)} = (p_{ij}^{(1)})_{i,j=1}^{n}$ lässt sich als Markov-Matrix auffassen. Die Übergangswahrscheinlichkeit $p_{ij}^{(t)}$ von x_i zu x_j nach t Zeitschritten ist dann der Eintrag an der Stelle (i, j) der Matrix $P^{(t)} := (P^{(1)})^t$. Die Anzahl der betrachteten Zeitschritte t ist als Parameter im Vorhinein festzusetzen.

Setzt man noch $\psi(x_i)^{(0)} := \frac{m_i}{\sum_{j=1}^{n} m_j}$, wobei $m_i := \sum_{j=1}^{n} p_{ij}^{(t)}$ der Grad eines Knotens x_i im Graph G ist, so lässt sich die Diffusionsdistanz $D^{(t)}(x_i, x_j)$ zwischen x_i und x_j definieren als

$$D^{(t)}(x_i, x_j) = \sqrt{\sum_{k=1}^{n} \frac{(p_{ik}^{(t)} - p_{jk}^{(t)})^2}{\psi(x_k)^{(0)}}}.$$

Der Ausdruck $\psi(x_k)^{(0)}$ im Nenner sorgt dafür, dass Teile des Datensatzes mit hoher Dichte für die Diffusionsdistanz stärker gewichtet werden (siehe [91]). Die Diffusionsdistanz $D^{(t)}(x_i, x_j)$ ist demnach klein für Punkte x_i und x_j mit hoher Übergangswahrscheinlichkeit. Die Diffusionsdistanz gilt als robuster gegenüber Störungen als etwa die geodätische Distanz. Den bei der Approximation der geodätischen Distanz in Isomap möglichen „Kurzschlüssen“ wird entgegengewirkt, da die Diffusionsdistanz auf vielen unterschiedlichen Pfaden zwischen x_i und x_j beruht (siehe [91]).

Es lässt sich zeigen (siehe Satz 1.26), dass die euklidische niedrigdimensionale Darstellung Y, welche die Diffusionsdistanz nahezu erhält, durch

$$Y = (y_1, \ldots, y_n)^T = (\lambda_2^t v_2, \ldots, \lambda_{d+1}^t v_{d+1}),$$

gegeben ist, wobei $\lambda_2^t, \ldots, \lambda_{d+1}^t$ die d größten, nicht trivialen Eigenwerte der Matrix $P^{(t)}$ und $v_2, \ldots, v_{d+1}$ die zugehörigen Eigenvektoren sind (vgl. [91]). Bezeichnet $v_l^{(i)}$ für $i = 1, \ldots, n$ und $l = 2, \ldots, d+1$ die i-te Komponente des Eigenvektors v_l, so wählt man demnach

$$y_i = (\lambda_2^t v_2^{(i)}, \ldots, \lambda_{d+1}^t v_{d+1}^{(i)})^T. \tag{1.4}$$

Gibt man eine Genauigkeit $\delta > 0$ vor und definiert

$$d := d(\delta, t) := \max\{l \in \mathbb{N} : |\lambda_l|^t > \delta |\lambda_2|^t\},$$

so erhält man eine Schätzung für die intrinsische Dimension d des Datensatzes. Die Isometrie im folgenden Satz 1.26 (vgl. [33]) rechtfertigt die Wahl der Einbettung gemäß (1.4).

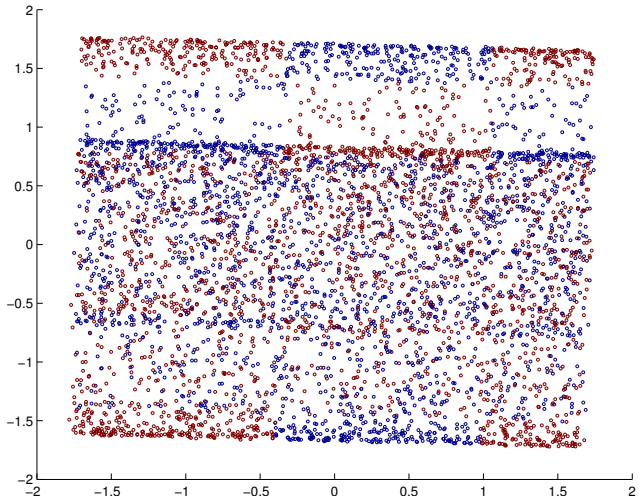

Abbildung 1.8: Mittels der Toolbox [89] erhaltene zweidimensionale Darstellung der Schweizer Rolle gemäß der Diffusionsabbildungen.

Satz 1.26 *Mit der Zieldimension $d = d(\delta, t)$ und y_i wie in (1.4) gilt für $i, j = 1, \ldots, n$*

$$\|y_i - y_j\|_{\mathbb{R}^d} = D^{(t)}(x_i, x_j)$$

bis auf relative Genauigkeit $\delta > 0$.

Bemerkung 1.27

1. *Die Diffusionsabbildungen erfordern eine Spektralzerlegung der Matrix $P^{(t)}$. Damit beträgt die Größenordnung der arithmetischen Komplexität der Diffusionsabbildungen $\mathcal{O}(n^3)$ und die des Speicherplatzbedarfs $\mathcal{O}(n^2)$ (siehe [91]).*

2. *Der Fokus der Diffusionsabbildungen liegt wie bei allen globalen Methoden auf der Erhaltung großer Abstände (hier in Form der Diffusionsdistanzen), wohingegen lokale Strukturen eine untergeordnete Rolle spielen.*

Eine exemplarische niedrigdimensionale Darstellung der Schweizer Rolle aus Abbildung 1.2 gemäß der Diffusionsabbildungen findet man in Abbildung 1.8. Dabei wurden die Parameter $t = 1$ und $\sigma = 1$ gewählt. Die Darstellung erinnert in diesem Fall an die mittels PCA (siehe Abbildung 1.3) gefundene und enthält noch Artefakte der Rollenstruktur.

1.2.2 Lokale Spektralmethoden

Lokal lineare Einbettung (LLE)

Ziel der lokal linearen Einbettung (engl. Locally Linear Embedding, LLE) [114, 117] ist eine Dimensionsreduktion unter Bewahrung der lokalen Geometrie, d.h. der Winkel und Nachbarschaften, der gegebenen hochdimensionalen Punkte. Gesucht ist demnach eine konforme, d.h. eine (lokal) winkeltreue, Abbildung der hochdimensionalen Punkte in den niedrigdimensionalen Raum. Dabei geht man in zwei Schritten vor:

1. Approximiere jeden Punkt x_i durch eine Linearkombination seiner Nachbarn mit sogenannten Rekonstruktionsgewichten w_{ij}, d.h.

$$x_i \approx \sum_{j=1}^{n} w_{ij} x_j$$

   mit $w_{ij} = 0$, falls $x_j \notin N(x_i)$, und der Normalisierung $\sum_{j=1}^{n} w_{ij} = 1$ für alle $i = 1, \ldots, n$. Es handelt sich also um eine Affinkombination der Nachbarn, jedoch im Allgemeinen nicht um eine Konvexkombination, da nicht notwendigerweise $0 \leq w_{ij} \leq 1$ für alle $i, j = 1, \ldots, n$ gilt.

2. Finde eine Einbettung, d.h. Punkte $y_1, \ldots, y_n \in \mathbb{R}^d$, die die Rekonstruktionsgewichte w_{ij} entsprechender Nachbarn optimal erhalten.

Dem Wunsch, jeden Datenpunkt im ersten Schritt durch seine k nächsten Nachbarn darstellen zu können, liegt die Annahme zugrunde, dass die Mannigfaltigkeit lokal linear ist (vgl. [84], S.152). Dies ist im Allgemeinen nur approximativ der Fall. Zur Konstruktion der Matrix $W = (w_{ij})_{i,j=1}^{n}$ der Rekonstruktionsgewichte ist daher das folgende restringierte Minimierungsproblem zu lösen.

Problem 1.28 *Finde* $W = (w_{ij})_{i,j=1}^{n} \in \mathbb{R}^{n \times n}$, *sodass*

$$\begin{aligned} \varepsilon(W) := \sum_{i=1}^{n} \|x_i - \sum_{j=1}^{n} w_{ij} x_j\|_2^2 &\to \min, \\ \sum_{j=1}^{n} w_{ij} = 1 \quad \forall i = 1, \ldots, n, & \\ w_{ij} = 0, \quad \textit{falls } x_j \notin N(x_i). & \end{aligned}$$

Man beachte, dass die erhaltenen Rekonstruktionsgewichte invariant unter Skalierung, Translation und Rotation sind (siehe [91]). Wenn $x_{i_1}, \ldots, x_{i_k}$ die Nachbarn von x_i sind

und $w_i = (w_{ii_1}, \ldots, w_{ii_k})$ der Vektor der zugehörigen Gewichte ist, lässt sich das Fehlerfunktional ε umschreiben zu

$$\varepsilon(W) = \sum_{i=1}^{n} \varepsilon_i(w_i)$$

mit

$$\begin{aligned}
\varepsilon_i(w_i) &= \|x_i - \sum_{j=1}^{k} w_{ii_j} x_{i_j}\|_2^2 \\
&= \|\sum_{j=1}^{k} w_{ii_j}(x_i - x_{i_j})\|_2^2 \\
&= \sum_{j=1}^{k}\sum_{l=1}^{k} w_{ii_j} w_{ii_l} \underbrace{\langle x_i - x_{i_j}, x_i - x_{i_l}\rangle}_{=:g_{jl}(x_i)} \\
&= \sum_{j=1}^{k}\sum_{l=1}^{k} w_{ii_j} w_{ii_l} g_{jl}(x_i).
\end{aligned}$$

Dabei folgt die zweite Gleichheit wegen $\sum_{j=1}^{k} w_{ii_j} = 1$. Die Matrix $\mathfrak{G}(x_i) = (g_{jl})_{j,l=1}^{k} = (\langle x_i - x_{i_j}, x_i - x_{i_l}\rangle)_{j,l=1}^{k}$ stellt eine Art lokale Gram-Matrix dar. Wenn $\mathfrak{G}(x_i)$ invertierbar ist, lässt sich zeigen, dass $\varepsilon_i(w_i)$ unter der Nebenbedingung $\sum_{j=1}^{k} w_{ii_j} = 1$ genau ein Minimum mit

$$w_{ii_j} = \frac{\sum_{l=1}^{k} h_{jl}(x_i)}{\sum_{l,m=1}^{k} h_{lm}(x_i)}$$

hat, wobei $\mathfrak{G}^{-1}(x_i) = (h_{lm}(x_i))_{l,m=1}^{k}$ die Inverse von $\mathfrak{G}(x_i)$ sei. Der Fall, dass $\mathfrak{G}(x_i)$ nicht invertierbar ist, wird in der Numerik durch eine passende Regularisierung umgangen (siehe [117]).

Zur Bestimmung der Einbettung im zweiten Schritt der LLE betrachtet man die folgende restringierte Optimierungsaufgabe.

Problem 1.29 *Finde $Y \in \mathbb{R}^{n\times d}$, sodass*

$$\begin{aligned}
\phi(Y) := \sum_{i=1}^{n} \|y_i - \sum_{j=1}^{n} w_{ij} y_j\|_2^2 &\to \min, \\
\sum_{i=1}^{n} y_i &= 0, \\
\frac{1}{n}\sum_{i=1}^{n} y_i y_i^T &= I.
\end{aligned}$$

Dabei bezeichne $I \in \mathbb{R}^{d \times d}$ die Einheitsmatrix. Grund für die Einführung der Nebenbedingungen ist die erwähnte Invarianz der Rekonstruktionsgewichte unter Skalierung, Translation und Rotation. Die erste Nebenbedingung beseitigt die Translationsinvarianz der Konfiguration $Y = (y_1, \ldots, y_n)^T$, die zweite Nebenbedingung die Skalierungsinvarianz bis auf Vorzeichen. Weiterhin unterdrückt die zweite Nebenbedingung die triviale Lösung $Y = 0$ (vgl. [91]).

Man kann zeigen, dass die Lösung des Problems im zweiten Schritt der LLE durch die Eigenvektoren $v_2, \ldots, v_{d+1}$ zu den d kleinsten von Null verschiedenen Eigenwerten der Matrix $(I - W)^T(I - W)$ gegeben ist. Für die gesuchte niedrigdimensionale Darstellung $Y = (y_1, \ldots, y_n)^T$ wählen wir demnach

$$Y = (v_2, \ldots, v_{d+1}).$$

Der kleinste Eigenwert von $(I - W)^T(I - W)$ ist Null mit zugehörigem Eigenvektor $v_1 = (1, \ldots, 1) \in \mathbb{R}^d$ und wird somit für die niedrigdimensionale Darstellung der Daten nicht verwendet (siehe [117]).

Bemerkung 1.30 *1. Man beachte, dass hier im Gegensatz zu den globalen Spektralmethoden die Spektralzerlegung einer dünnbesetzten Matrix erfolgt. Ist p das Verhältnis von Einträgen ungleich Null zur Gesamtzahl der Einträge der Matrix $(I - W)^T(I - W)$, so liegen die arithmetische Komplexität und der Speicherplatzbedarf der LLE in der Größenordnung $\mathcal{O}(pn^2)$ (siehe [91]).*

2. *Da die LLE nur lokale Eigenschaften der zu lernenden Mannigfaltigkeit nutzt, leidet sie weniger unter Problemen wie den „Kurzschlüssen“ bei Isomap. Im Gegenzug werden globale Eigenschaften zum Teil unzureichend in der niedrigdimensionalen Darstellung widergespiegelt (vgl. [91]).*

3. *Kritisch ist die Feinabstimmung der freien Parameter. Die passende Wahl des Nachbarschaftsparameters k sowie eines Regularisierungsparameters, welcher bei der numerischen Bestimmung der Rekonstruktionsgewichte benötigt wird, beeinflusst die resultierende Einbettung maßgeblich (vgl. [84], S.157).*

4. *Eine Verallgemeinerung einer mit der LLE bestimmten niedrigdimensionalen Darstellung (Out-of-sample-Erweiterung) lässt sich auf verschiedene Wege erhalten (siehe [84], S.156 f.). Der erste, in [117] beschriebene, Weg führt über eine lokal lineare Interpolation. Der zweite, ebenso in [117] vorgeschlagene, Weg lernt ein künstliches neuronales Netz auf die niedrigdimensionale Darstellung gemäß der LLE ein. Auf diese Weise erhält man ein parametrisches Modell und kann neue Punkte einbetten. Einen dritten Weg, der über die Nyström-Formel und die Interpretation der LLE als Kern-Methode führt, findet man in [16].*

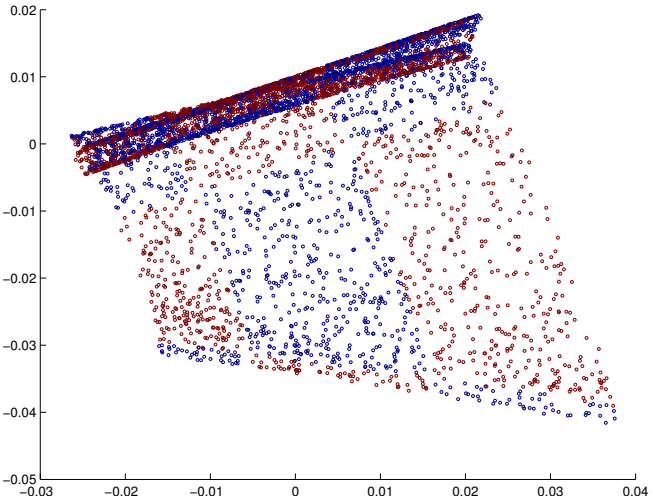

Abbildung 1.9: Mittels der Toolbox [89] erhaltene zweidimensionale Darstellung der Schweizer Rolle gemäß LLE.

In Abbildung 1.9 sieht man eine niedrigdimensionale Darstellung der Schweizer Rolle aus Abbildung 1.2 gemäß der LLE. Für die Nachbarschaftsdefinition wurden die $k = 12$ nächsten Nachbarn betrachtet. Die LLE erkennt die intrinsische Geometrie der Schweizer Rolle gut. Negativ ist, dass ein relativ großer Teil der Datenpunkte auf einen relativ schmalen Streifen am oberen Ende abgebildet wird.

Laplacesche Eigenabbildungen (Laplacian Eigenmaps)

Die Laplaceschen Eigenabbildungen (engl. Laplacian Eigenmaps) [12, 13] nutzen aus, dass die Eigenwerte und Eigenvektoren des Graph-Laplace-Operators (engl. graph Laplacian) Aufschluss über Eigenschaften des zugehörigen Graphen, d.h. hier über lokale Informationen der Mannigfaltigkeit, geben (vgl. [84], S.159 und [28]).

Die Methode der Laplaceschen Eigenabbildungen beruht auf der Konstruktion eines nicht gerichteten, gewichteten Nachbarschaftsgraphen G der Daten $X = (x_1, \ldots, x_n)^T$, in dem jeder Punkt x_i mit seinen k nächsten Nachbarn verbunden ist. Für die Kantengewichte w_{ij} benutzt man etwa einen Gauß-Kern, d.h.

$$w_{ij} = \exp\left(-\frac{\|x_i - x_j\|_2^2}{t}\right) \quad \text{für } (i,j) \in G$$

mit einem Parameter $t > 0$. Dies lässt sich als Wärmeleitungskern mit Temperaturparameter t interpretieren (siehe [84], S.160). Kanten von nah zusammenliegenden Punkten erhalten demnach relativ große Gewichte (nahe 1), Kanten von weit auseinanderliegenden Punkten relativ kleine Gewichte (nahe 0). Durch die Gewichte ist ein

Ähnlichkeitsmaß für die Datenpunkte gegeben. Als Grenzfall $t = \infty$ ist auch eine triviale Gewichtsfunktion mit

$$w_{ij} = 1 \quad \text{für } (i,j) \in G$$

denkbar. Man kann den Nachbarschaftsgraphen auch als vollständigen Graphen (d.h., alle Punkte sind untereinander mit Kanten verbunden), in dem die Kantengewichte zwischen nicht benachbarten Punkten Null sind, definieren.

Um die niedrigdimensionale Darstellung $Y = (y_1, \ldots, y_n)^T$ zu erhalten minimieren wir nun das Funktional

$$\Phi(Y) = \sum_{i,j=1}^{n} \|y_i - y_j\|_2^2 w_{ij}. \tag{1.5}$$

Offenbar findet man triviale Lösungen, indem man alle y_i identisch wählt, z.B. $Y = 0$. Zur Vermeidung dieser unerwünschten Lösungen fügen wir später noch eine Nebenbedingung hinzu. Die Gewichtung der Abstandsquadrate $\|y_i - y_j\|^2$ in (1.5) sorgt dafür, dass zwei Punkte y_i und y_j in der niedrigdimensionalen Darstellung möglichst nahe zusammen liegen, wenn die entsprechenden Punkte x_i und x_j der gegeben hochdimensionalen Daten nahe zusammen liegen. Diese Minimierung der lokalen Abstände soll die Nachbarschaften und somit die Topologie der Mannigfaltigkeit erhalten (vgl. [84], S.160).

Die Lösung des obigen Minimierungsproblem steht in engem Zusammenhang mit dem Graph-Laplace-Operator. Der Graph-Laplace-Operator L ist mit $W = (w_{ij})_{i,j=1}^n$, $m_i = \sum_{j=1}^n w_{ij}$ und $M = \operatorname{diag}(m_1, \ldots, m_n)$ definiert als

$$L = M - W.$$

Damit gilt nun (vgl. [84], S.161), wenn $y_i^{(p)}$ für $p = 1, \ldots, d$ die p-te Komponente von y_i bezeichnet,

$$\begin{aligned}
\Phi(Y) &= \sum_{i,j=1}^{n} \|y_i - y_j\|_2^2 w_{ij} \\
&= \sum_{p=1}^{d} \sum_{i,j=1}^{n} (y_i^{(p)} - y_j^{(p)})^2 w_{ij} \\
&= \sum_{p=1}^{d} \left(\sum_{i=1}^{n} (y_i^{(p)})^2 \underbrace{\sum_{j=1}^{n} w_{ij}}_{=m_i} + \sum_{j=1}^{n} (y_j^{(p)})^2 \underbrace{\sum_{i=1}^{n} w_{ij}}_{=m_j} - 2 \sum_{i,j=1}^{n} y_i^p y_j^{(p)} w_{ij} \right) \\
&= 2 \sum_{p=1}^{d} (\underbrace{\sum_{i=1}^{n} (y_i^{(p)})^2 m_i}_{=(Y^T M Y)_{pp}} - \underbrace{\sum_{i,j=1}^{n} y_i^{(p)} y_j^{(p)} w_{ij}}_{=(Y^T W Y)_{pp}}) \\
&= 2 \operatorname{trace}(Y^T (M - W) Y) \\
&= 2 \operatorname{trace}(Y^T L Y).
\end{aligned}$$

Nun führt man die Nebenbedingung $Y^T M Y = I$ ein, um die erwähnten trivialen Lösungen zu unterdrücken. Die Laplaceschen Eigenabbildungen lösen dann folgendes restringierte Optimierungsproblem (siehe [91]).

Problem 1.31

$$\begin{aligned}
&\operatorname{trace}(Y^T L Y) \to \min \\
&Y^T M Y = I
\end{aligned}$$

Es lässt sich zeigen (siehe [12]), dass die Lösung dieser restringierten Minimierungsaufgabe den d Eigenvektoren v_i zu den d kleinsten von 0 verschiedenen Eigenwerten λ_i des verallgemeinerten Eigenwertproblems

$$L v_i = \lambda_i M v_i \tag{1.6}$$

entspricht. Die Matrix L ist symmetrisch und positiv semi-definit. Die Eigenwerte sind folglich reell und nicht negativ. Weiterhin verschwindet der kleinste Eigenwert λ_n mit zugehörigem Eigenvektor $v_n = (1, \ldots, 1) \in \mathbb{R}^n$ (siehe [12]).

Bemerkung 1.32 *1. Wie bei der LLE liegen die arithmetische Komplexität und der Speicherplatzbedarf der Laplacesche Eigenabbildungen in der Größenordnung*

$\mathcal{O}(pn^2)$, wobei p das Verhältnis von Einträgen ungleich Null zur Gesamtzahl der Einträge der Matrix L ist (siehe [91]).

2. *Interpretiert man den erstellten Graphen als diskrete Darstellung der Mannigfaltigkeit, so entspricht der Graph-Laplace-Operator einer diskreten Version des Laplace-Beltrami-Operators der Mannigfaltigkeit. Die errechneten Eigenvektoren als Lösungen von* (1.6) *kann man dann als diskrete Approximationen der Eigenfunktionen des Laplace-Beltrami-Operators auffassen (siehe [84], S.162).*

3. *In [13] wird gezeigt, dass die Laplaceschen Eigenabbildungen unter gewissen Voraussetzungen äquivalent mit der LLE sind. Weiterhin sind die Laplaceschen Eigenabbildungen mit den Diffusionsabbildungen verwandt, welche denselben Wärmeleitungskern nutzen.*

4. *Eine Möglichkeit, gemäß der Laplaceschen Eigenabbildungen gefundene niedrigdimensionalen Darstellungen mit Hilfe der Nyström-Formel auf neue Punkte zu verallgemeinern (Out-of-sample-Erweiterung), wird in [16] erwähnt.*

In der Praxis zeigen die Laplaceschen Eigenabbildungen leider oft unzureichende Resultate bei der Einbettung hochdimensionaler Daten. Zudem hat die Wahl der Parameter und der Gewichtsfunktion dramatischen Einfluss auf die Form der Einbettung (vgl. [84], S.163 f.). Die Laplaceschen Eigenabbildungen werden auch außerhalb der Dimensionsreduktion als Clustering-Methode [13] oder zur Partitionierung von Graphen [122] verwendet. Lee und Verleysen schlussfolgern (siehe [84], S.164), dass die Laplaceschen Eigenabbildungen besser für letztere Zwecke als zur Dimensionsreduktion geeignet sind.

In Abbildung 1.10 ist eine niedrigdimensionale Darstellung der Schweizer Rolle aus Abbildung 1.2 gemäß der Laplaceschen Eigenabbildungen dargestellt. Es wurde mit den $k = 12$ nächsten Nachbarn gearbeitet. Die „Entfaltung“ der Rolle kann nicht als gelungen bewertet werden, da die Daten auf einen schmalen gekrümmten Streifen komprimiert werden.

Auf der Hesse-Matrix basierende lokal lineare Einbettung (HLLE)

Eine Abwandlung der LLE stellt die von Donoho und Grimes vorgeschlagene auf der Hesse-Matrix basierende lokal lineare Einbettung (engl. Hessian-based Locally Linear Embedding, HLLE) [46] dar. Wie die Laplaceschen Eigenabbildungen beruht die HLLE auf der Analyse eines Differentialoperators. Während die Laplaceschen Eigenabbildungen den Laplace-Beltrami-Operator betrachten, nutzt die HLLE die Hesse-Matrix (vgl. [91]). Donoho und Grimes formulieren den theoretischen Rahmen der HLLE im kontinuierlichen Fall und betrachten glatte Riemannsche Mannigfaltigkeiten M.

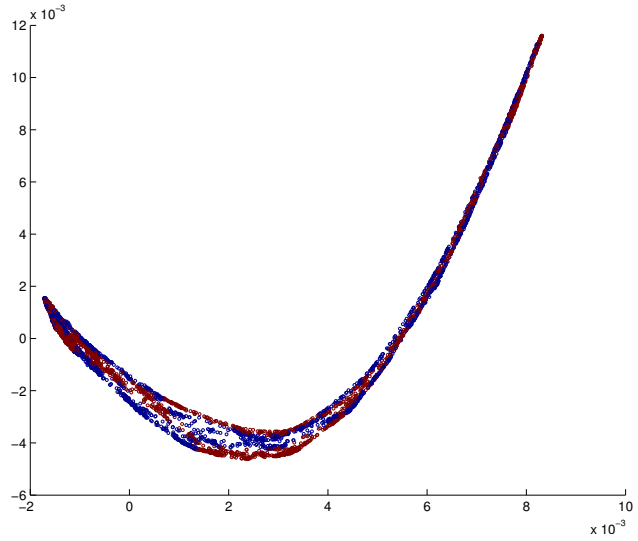

Abbildung 1.10: Mittels der Toolbox [89] erhaltene zweidimensionale Darstellung der Schweizer Rolle gemäß der Laplaceschen Eigenabbildungen.

Für die niedrigdimensionale Einbettung wird eine Minimierung der Krümmung der Mannigfaltigkeit unter der Annahme, dass der Parameterraum lokal isometrisch zur hochdimensionalen Mannigfaltigkeit ist, angestrebt (siehe [91]). Dies erreicht man mittels einer Analyse des für Funktionen $f : M \to \mathbb{R}$ definierten Operators

$$\mathcal{H}(f) = \int_M \|H_f(m)\|_F^2 \mathrm{d}m,$$

der die quadrierte Frobenius-Norm der Hesse-Matrix $H_f(m)$ über die Mannigfaltigkeit M integriert. Dabei ist $\mathrm{d}m$ ein passendes Wahrscheinlichkeitsmaß mit positiver Dichte im Inneren von M. Die Hesse-Matrix $H_f(m)$ wird bezüglich orthogonaler Koordinaten im lokalen Tangentialraum von $m \in M$ dargestellt, wobei f in einer Umgebung von m zweimal stetig differenzierbar sei.

Donoho und Grimes beweisen, dass man den offenen, zusammenhängenden Parameterraum $\Theta \subset \mathbb{R}^d$ einer Mannigfaltigkeit $M = \psi(\Theta)$, wobei $\psi : \Theta \to \mathbb{R}^D$ eine lokal isometrische Einbettung darstellt, bis auf Rotation und Translation durch eine passende Basis des Nullraumes des Operators $\mathcal{H}$ bestimmen kann (siehe [46]). Diese theoretische Optimalitätsgarantie ist, da Konvexität der Mannigfaltigkeit hierfür nicht notwendig ist, gewissermaßen stärker als die von Isomap (siehe [28]).

Im diskreten Fall bestimmt man zunächst die k nächsten Nachbarn bezüglich euklidischer Distanz jedes Datenpunktes x_i, $i = 1, \ldots, n$, und errechnet eine Basis des lokalen Tangentialraumes an x_i mittels einer auf die Menge der k nächsten Nachbarn des Da-

tenpunktes x_i angewandten PCA (vgl. [91]). Die Matrix $\mathfrak{H} = (\mathfrak{H}_{l,m})_{l,m=1}^{n}$ als diskrete Version des Operators $\mathcal{H}$ ist dann durch

$$(\mathfrak{H})_{l,m} = \sum_{i,j=1}^{n} (H_i)_{j,l}(H_i)_{j,m}$$

gegeben. Dabei sind H_i Matrizen, welche sich als diskrete Approximationen der Hesse-Matrizen an x_i in Koordinaten des lokalen Tangentialraumes auffassen lassen. Durch die Eigenvektoren zu den $d+1$ kleinsten Eigenwerten von $\mathfrak{H}$ ist eine Approximation des Nullraumes von $\mathcal{H}$ gegeben. Die gesuchte niedrigdimensionale Darstellung der Mannigfaltigkeit erhält man als die Basisvektoren einer passenden Basis des Untervektorraumes, der von den Eigenvektoren, die zu den d kleinsten von Null verschiedenen Eigenwerten von $\mathfrak{H}$ gehören, aufgespannt wird (siehe [46]).

Damit ergibt sich für die HLLE analog zur LLE und den Laplaceschen Eigenabbildungen eine Größenordnung der arithmetischen Komplexität und des Speicherplatzbedarfs von $\mathcal{O}(pn^2)$, wobei p das Verhältnis von Einträgen ungleich Null zur Gesamtzahl der Einträge der Matrix $\mathfrak{H}$ bezeichnet (siehe [91]).

Aufgrund der Nähe der HLLE zur LLE und den Laplaceschen Eigenabbildungen, hat die HLLE mit ähnlichen Schwächen wie diese zwei Methoden zu kämpfen (siehe [91]). Donoho und Grimes geben in [46] auch zu bedenken, dass Approximationen zweiter Ableitungen von Funktionen (innerhalb der Hesse-Matrix) in sehr hohen Dimensionen schwierig und störanfällig sein können.

Abbildung 1.5 stellt eine durch die HLLE berechnete niedrigdimensionale Darstellung der Schweizer Rolle aus Abbildung 1.2 dar. Als Nachbarn eines Punktes wurden die $k = 12$ nächsten Punkte angesehen. In diesem Beispiel ermittelt die HLLE die intrinsische Struktur der Mannigfaltigkeit sehr gut.

Bemerkung 1.33 *Ein anderer Ansatz einer lokalen Spektralmethode zur Dimensionsreduktion, welche lokale Eigenschaften des hochdimensionalen Datensatzes wie die HLLE mit Hilfe der lokalen Tangentialräume erfasst (vgl. [91]), ist die Anpassung lokaler Tangentialräume (engl. Local Tangent Space Alignment, LTSA) [144], auf die wir hier aber nicht genauer eingehen möchten.*

1.2.3 Abstandserhaltende Nicht-Spektralmethoden

Im Folgenden werden Dimensionsreduktionsmethoden vorgestellt, welche auf eine Abstandserhaltung durch Minimierung gewisser Kostenfunktionale abzielen. Die entstehenden Optimierungsprobleme sind im Allgemeinen nicht konvex und lassen sich im

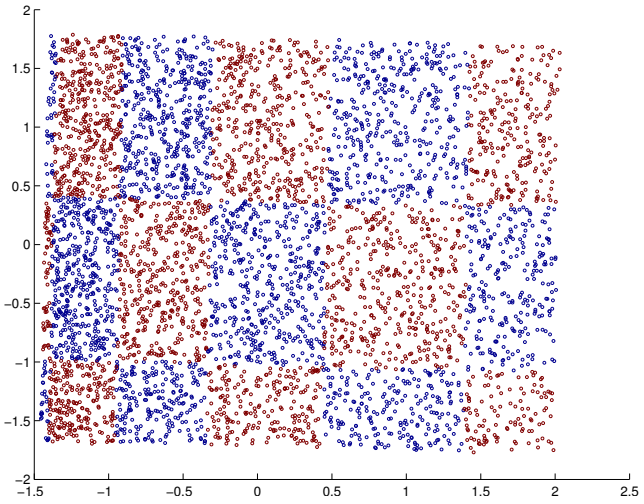

Abbildung 1.11: Mittels der Toolbox [89] erhaltene zweidimensionale Darstellung der Schweizer Rolle gemäß HLLE.

Gegensatz zu den bisher vorgestellten Methoden nicht mittels der Spektralzerlegung einer Matrix lösen. Stattdessen werden spezielle Optimierungstechniken benötigt.

Sammon-Abbildung (Sammon Mapping)

Die Sammon-Abbildung (engl. Sammon mapping) wurde von Sammon [116] vorgeschlagen und in der Folge weiterentwickelt, z.B. in [99]. Die Methode stellt eine Variante der MDS dar. Ziel ist eine Erhaltung der (euklidischen) Distanzen in der zu bestimmenden niedrigdimensionalen Darstellung. Statt des bei der MDS verwendeten Kostenfunktionals minimiert man jedoch das Sammon-Kostenfunktional

$$\phi(Y) = \frac{1}{\sum_{i,j=1}^{n} \|x_i - x_j\|_2} \sum_{\substack{i,j=1 \\ i \neq j}}^{n} \frac{(\|x_i - x_j\|_2 - \|y_i - y_j\|_2)^2}{\|x_i - x_j\|_2}.$$

Durch das Sammon-Kostenfunktional werden bevorzugt die paarweisen Abstände der hochdimensionalen Punkte $x_1, \ldots, x_n \in \mathbb{R}^D$ in der niedrigdimensionalen Darstellung $y_1, \ldots, y_n \in \mathbb{R}^d$ beibehalten, die ursprünglich klein sind. Durch diese Gewichtung soll die lokale Geometrie besser erfasst werden als bei der MDS, die bevorzugt große Abstände erhält (siehe [91]). Die Sammon-Abbildung kann daher auch nicht lineare Mannigfaltigkeiten erkennen, sofern sie nicht zu stark verformt sind (siehe [84], S.86). Die Distanzen $\|x_i - x_j\|_2$ im Nenner der Summe des Sammon-Kostenfunktionals können jedoch auch problematisch sein, falls unter ihnen sehr kleine Werte vorkommen (vgl. [91]).

Das zu lösende Minimierungsproblem ist nicht konvex. Es existiert keine Lösung in geschlossener Form. Zur Bestimmung einer Lösung verwendet man eine Quasi-Newton-Methode. Hierbei ist Vorsicht geboten, da der Algorithmus in einem lokalen Minimum stoppen kann. Weiterhin kann die Konvergenz gegen das Minimum für bestimmte Datensätze langsam sein. Werden T Iterationen der Quasi-Newton-Methode durchgeführt, so liegt die arithmetische Komplexität der Sammon-Abbildung in der Größenordnung $\mathcal{O}(Tn^2)$. Der Speicherplatzbedarf ist von der Größe $\mathcal{O}(n^2)$ (siehe [91]).

Für ein Beispiel einer niedrigdimensionalen Darstellung der Schweizer Rolle aus Abbildung 1.2 gemäß der Sammon-Abbildung sei auf Abbildung 1.12 verwiesen. Hier gelingt die „Entfaltung“ nur bedingt und die Rollenstruktur lässt sich aufgrund von Überlagerungen noch erahnen.

Bemerkung 1.34 *1. Die Sammon-Abbildung ist eine oft angewandte Methode, auch weil einige Varianten, die die aus der Nichtkonvexität des Kostenfunktionals und der Verwendung der vollen Distanzmatrix resultierenden Nachteile reduzieren, eingeführt wurden (vgl. [84], S.86 f.), z.B. die Sammon-Abbildung in Verbindung mit künstlichen neuronalen Netzen (engl. Sammon artificial neural networks, SAMANN) [111] oder die Triangulierungsansätze in [99].*

2. Weiterhin existieren auch unter dem Kürzel GNLM (engl. geodesic nonlinear mapping) bekannte Abwandlungen der Sammon-Abbildung, die wie Isomap Graph-Distanzen als Approximationen der geodätischen Distanzen anstelle der euklidischen Abstände $\|x_i - x_j\|_2$ nutzen (siehe etwa [100, 142]). Diese Varianten erzeugen für stärker gekrümmte Mannigfaltigkeiten bessere niedrigdimensionale Darstellungen als die Sammon-Abbildung auf Grundlage euklidischer Distanzen (siehe [84], S.113 f.).

Kurvilinearkomponentenanalyse (CCA)

Die Kurvilinearkomponentenanalyse (engl. Curvilinear Component Analysis, CCA) wurde von Demartines und Hérault [42, 43] als eine Erweiterung der Selbstorganisierenden Abbildungen (SOM) von Kohonen, die wir in Unterabschnitt 1.2.4 noch thematisieren, vorgeschlagen. Wie bei den SOM greift man bei der CCA auf Optimierungstechniken aus dem Gebiet der künstlichen neuronalen Netzwerke zurück und kombiniert Vektorquantisierung (siehe Unterabschnitt 1.1.2) und Dimensionsreduktion. Die Vektorquantisierung ist in den aktuellen Versionen der CCA jedoch nur noch optional. Vorgängerversionen der CCA enthielten eine Vektorquantisierung noch als festen Bestandteil und waren daher auch unter dem Namen „Vektorquantisierung und Projektion“ (engl. Vector Quantization and Projection, VPQ) [41] bekannt. Die CCA ist eine abstandserhaltende, nicht li-

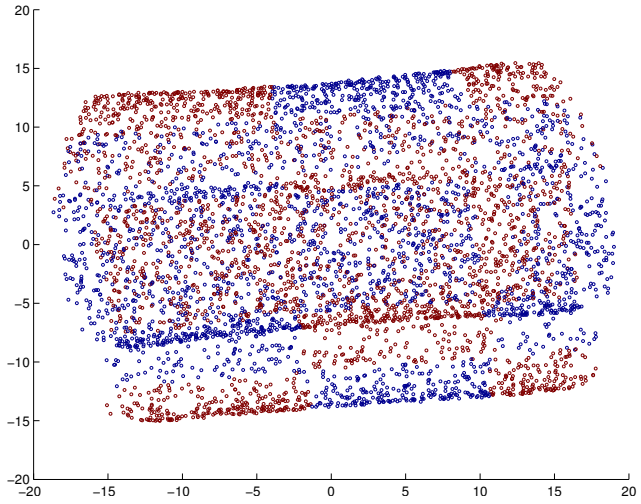

Abbildung 1.12: Mittels der Toolbox [89] erhaltene zweidimensionale Darstellung der Schweizer Rolle gemäß der Sammon-Abbildung.

neare Dimensionsreduktionsmethode, welche Parallelen zur zuvor vorgestellten Sammon-Abbildung zeigt (siehe [84], S.88).

Die CCA minimiert das Fehlerfunktional

$$E(Y) = \frac{1}{2}\sum_{i,j=1}^{n}(\|x_i - x_j\|_D - \|y_i - y_j\|_d)^2 F_\lambda(\|y_i - y_j\|_D)$$

mit einer positiven, monoton fallenden Funktion $F_\lambda : \mathbb{R} \to \mathbb{R}$ mit einem Parameter $\lambda > 0$. Dieses Fehlerfunktional weist große Ähnlichkeit zum bei der Sammon-Abbildung verwendeten Funktional auf. Durch die Gewichtung mit $F_\lambda(\|y_i - y_j\|_D)$ werden kurze Abstände bevorzugt gegenüber längeren erhalten, um einerseits die globale Form der Mannigfaltigkeit zu „entfalten“ und andererseits lokale Strukturen zu erhalten (vgl. [84], S.89). Man beachte, dass die Gewichtung im Gegensatz zur Sammon-Abbildung jedoch nicht von den a priori bekannten hochdimensionalen paarweisen Abständen abhängt, sondern von den zu bestimmenden niedrigdimensionalen, welche innerhalb eines iterativen Algorithmus von Iteration zu Iteration veränderlich sind (vgl. [84], S.89).

Die Minimierung von $E(Y)$ erfolgt mittels eines Gradientenverfahrens. Da ein übliches Gradientenverfahren in diesem Fall zu langsam konvergiert, haben Demartines und Hérault eine Modifikation entworfen, bei der das Fehlerfunktional in $E(Y) = \sum_{i=1}^{n} E^i(Y)$ mit

$$E^i(Y) = \frac{1}{2}\sum_{j=1}^{n}(\|x_i - x_j\|_D - \|y_i - y_j\|_d)^2 F_\lambda(\|y_i - y_j\|_D)$$

aufgeteilt wird und die einzelnen Summanden $E^1(Y), \ldots, E^n(Y)$ separat gemäß Gradientenverfahren iterativ optimiert werden (siehe [84], S.90).

Für ein festes i lässt sich die Aktualisierungsvorschrift

$$y_j \mapsto y_j - \alpha \nabla_{y_j} E^i = y_j - \alpha\beta(i,j)\frac{y_i - y_j}{\|y_i - y_j\|_d} \tag{1.7}$$

für $j \in \{1,\dots,n\} \setminus \{i\}$ mit einer sogenannten Lernrate $\alpha > 0$ ableiten (vgl. [84], S.90). Dabei ist

$$\beta(i,j) = [d_x(i,j) - d_y(i,j)][2F_\lambda(d_y(i,j)) - (d_x(i,j) - d_y(i,j))F'_\lambda(d_y(i,j))]$$

mit den Bezeichnungen $d_x(i,j) = \|x_i - x_j\|_D$ und $d_y(i,j) = \|y_i - y_j\|_d$. Für die Abstandserhaltung ist wünschenswert, dass y_i und y_j durch die Aktualisierung näher zusammenrücken, wenn $d_x(i,j) - d_y(i,j) < 0$ ist. Folglich sollte

$$2F_\lambda(d_y(i,j)) > (d_x(i,j) - d_y(i,j))F'_\lambda(d_y(i,j)) \tag{1.8}$$

gelten (siehe [84], S.90).

Ein Schritt des iterativen Algorithmus wird Epoche genannt. In jeder Epoche wird ein Punkt y_{i_1} mit $i_1 \in \{1,\dots,n\}$ fixiert und die restlichen Punkte y_j für $j \in \{1,\dots,n\}\setminus\{i_1\}$ werden gemäß (1.7) aktualisiert. Dann wird ein Punkt y_{i_2} mit $i_2 \in \{1,\dots,n\}\setminus\{i_2\}$ fixiert und wiederum werden die restlichen Punkte y_j für $j \in \{1,\dots,n\}\setminus\{i_2\}$ aktualisiert. Dies wird fortgeführt, bis jeder Punkt $y_1,\dots,y_n$ genau einmal fixiert war. Danach geht man mit reduzierten Werten für die Parameter α und λ zur nächsten Epoche über, sofern noch keine Konvergenz vorliegt.

Der Algorithmus der CCA lässt sich nun wie folgt zusammenfassen (vgl. [84], S.93).

Algorithmus 1.35 *1. Vektorquantisierung (optional)*

2. *Berechne alle paarweisen Abstände $\|x_i - x_j\|_D$ für $i,j = 1,\dots,n$.*

3. *Initialisiere die d-dimensionalen Punkte $y_1,\dots,y_n$ und Epochennummer $q = 1$.*

4. *Lege die Parameter α und λ für die Epoche q fest.*

5. *Fixiere einen Punkt y_i und aktualisiere die übrigen Punkte y_j für $j \in \{1,\dots,n\}\setminus\{i\}$ gemäß (1.7).*

6. *Wiederhole Schritt 5, bis alle y_i genau einmal fixiert wurden.*

7. *Falls noch keine Konvergenz (im Sinne hinreichend kleiner Aktualisierungen der Punkte) vorliegt, setze $q \mapsto q+1$ und gehe zu Schritt 4.*

Bemerkung 1.36 *1. Die Initialisierung der d-dimensionalen Punkte y_i in Schritt 3 erfolgt entweder zufällig oder durch eine PCA.*

2. Die Wahl der Funktion F_λ ist wesentlich. Eine naheliegende Wahl ist z.B.

$$F_\lambda(u) = \exp\left(-\frac{u}{\lambda}\right).$$

Die Festlegung von λ, sodass (1.8) erfüllt ist, ist in diesem Fall jedoch schwierig (siehe [84], S.91 f.). Alternativ kann man $F_\lambda(u) = H(\lambda - u)$ mit der Heaviside-Funktion $H : \mathbb{R} \to \{0, 1\}$,

$$H(u) = \begin{cases} 0, & \text{falls } u \leq 0 \\ 1, & \text{falls } u > 0 \end{cases}$$

verwenden. Bei diesem Ansatz werden nur Abstände berücksichtigt, welche kleiner als λ sind. Der Parameter λ lässt sich somit als Nachbarschaftsweite interpretieren (siehe [84], S.92).

3. Das stochastische Gradientenverfahren konvergiert allgemein schneller als das bei der Sammon-Abbildung verwendete Quasi-Newton-Verfahren, kann jedoch ebenfalls in einem lokalen Minimum stoppen (siehe [84], S.95). Die Ordnung der arithmetischen Komplexität der CCA ist durch $\mathcal{O}(m^2 d)$ und die des Speicherplatzbedarfs durch $\mathcal{O}(m^2)$ gegeben (siehe [84], S.92), wobei m die Anzahl der aus der Vektorquantisierung resultierenden Prototypen (siehe Unterabschnitt 1.1.2) bezeichnet.

4. Für Zwecke einer Out-of-sample-Erweiterung ist die CCA in [43] mit einer gut funktionierenden Interpolationsprozedur ausgestattet, deren Erfolg jedoch unter einer zu hohen Ausgangsdimensionen oder dem Einfluss von Rauschen leiden kann (siehe [84], S.93).

Für ein Beispiel einer per CCA erstellten niedrigdimensionalen Darstellung einer Schweizer Rolle sei etwa auf [84](S.94) verwiesen. Die CCA deckt hier die intrinsische Geometrie der Rolle bei kaum vorhandenen Überlagerungen auf. Einige Nachbarschaftsbeziehungen von Datenpunkten werden jedoch fälschlich abgebildet.

Bemerkung 1.37 *1. In [67] wird von Hérault et al. eine Erweiterung der CCA vorgeschlagen, die insbesondere auch für verrauschte Datensätze nützlich ist (siehe [84], S.95 ff.). Hierbei wird das Verfahren in zwei simultane Teile zerlegt, globale „Entfaltung“ der Mannigfaltigkeit einerseits und Projektion der gestörten Daten auf die zugrundeliegende Mannigfaltigkeit andererseits. Dies sind gewissermaßen konkurrierende Ziele, da ersteres eine Verlängerung und letzteres eine Verkürzung paarweiser Abstände erfordert (siehe [84], S.95).*

2. *Venna und Kaski führen in [137] eine Art Kombination aus der CCA und der Sammon-Abbildung ein, indem sie die Funktion* $F_\lambda : \mathbb{R}^2 \to \mathbb{R}$,

$$F_\lambda(d_y(i,j), d_x(i,j)) = (1-\rho)H(\lambda - d_y(i,j)) + \rho H(\lambda - d_x(i,j))$$

nutzen. Dabei ist $\rho \in [0,1]$ *ein Parameter und* H *bezeichnet die zuvor definierte Heaviside-Funktion.*

3. *Eine bedeutende Weiterentwicklung der CCA stellt die Kurvilineardistanzenanalyse (Curvilinear Distance(s) Analysis, CDA) [82, 83] dar. Im Wesentlichen verhält sich die CDA zur CCA wie Isomap zur MDS. Für eine bessere Erfassung der Struktur der betreffenden Mannigfaltigkeit werden sogenannte Graph-Distanzen anstelle euklidischer Distanzen für die paarweisen Abstände* $d_x(i,j)$ *im Fehlerfunktional betrachtet. Die Graph-Distanzen werden wie in Isomap als Approximation der geodätischen Distanzen mittels des Algorithmus von Dijkstra [45] berechnet. Ein weitere Modifikation der CDA gegenüber der CCA betrifft die Gewichtungsfunktion* F_λ*. Der für die CCA übliche Ansatz einer einheitlichen Nachbarschaftsbreite* λ *führt zu unterschiedlich mächtigen Nachbarschaften der Punkte* y_i*. Bei der CDA gibt man statt der Nachbarschaftsbreite* λ *eine Nachbarschaftsproportion* $p \in [0,1]$ *an, welche angibt, wieviel Prozent der* n *Datenpunkte in der Nachbarschaft eines jeden Punktes* y_i *enthalten sein sollen. Es handelt sich demnach gewissermaßen um eine* k*-Nachbarschaft mit* $k \approx \lfloor pn \rfloor$*. Für jeden Datenpunkt* y_i *wird nun eine Nachbarschaftsbreite* λ_i *bestimmt, sodass die Kugel um* y_i *mit Radius* λ_i *ungefähr* $\lfloor pn \rfloor$ *Punkte enthält. Diese modifizierte Funktion* F_λ *macht die CDA robuster gegenüber Ausreißern als die CCA (siehe [84], S.114 ff.).*

Bemerkung 1.38 *Die Kurvilinearkomponentenanalyse (engl. Curvilinear Component Analysis, CCA) ist nicht zu verwechseln mit der grundverschiedenen Methode der konformen Eigenabbildungen (engl. Conformal Eigenmaps) [120], welche zuweilen auch mit „CCA“ (für engl. Conformal Component Analysis) abgekürzt wird. Die konformen Eigenabbildungen stellen eine Erweiterung der lokal linearen Einbettung (LLE) oder der Laplaceschen Eigenabbildungen dar, wobei eine niedrigdimensionale Darstellung unter bestmöglicher Erhaltung der Winkel innerhalb der gegebenen hochdimensionalen Daten angestrebt wird.*

1.2.4 Topologie erhaltende Nicht-Spektralmethoden

Im folgenden Unterabschnitt werden zwei Methoden zur Dimensionsreduktion dargestellt, welche wie die Methoden des vorherigen Unterabschnittes 1.2.3 nicht auf Spektraltechniken zurückgreifen. Im Unterschied zu den abstandserhaltenden Methoden in Unterabschnitt 1.2.3 liegt der Fokus bei den folgenden Methoden auf der Erhaltung der Topologie, d.h. der Erhaltung lokaler Strukturen wie Nachbarschaften von Datenpunkten.

Selbstorganisierende Abbildungen (SOM)

Einen gegenüber den bisher betrachteten Methoden völlig anderen Ansatz zur Dimensionsreduktion stellen die selbstorganisierenden Abbildungen (engl. Self-Organizing Maps, SOM) dar. Diese gehen auf von der Malsburg [95] zurück und wurden später durch die Weiterentwicklung in Kohonens Arbeit [75] populär. Sie sind daher auch unter dem Namen Kohonens selbstorganisierende Abbildungen (engl. Kohonen's Self-Organizing Maps, KSOM) bekannt (vgl. [84], S.135).

Die SOM führen eine besondere Form der Vektorquantisierung (siehe Unterabschnitt 1.1.2), bei der die Menge der Datenpunkte durch eine kleinere Menge von Repräsentanten, den sogenannten Prototypen, ersetzt wird, durch und verknüpfen dies mit einer Dimensionsreduktion. Bei den SOM wird ein d-dimensionales Gitter in Form von Punkten, den Prototypen, welche über Kanten mit ihren Nachbarn verbunden sind, vordefiniert. Diese Verbindungen der Prototypen innerhalb des Gitters sind fest, sodass die Prototypen im Unterschied zu üblichen Vektorquantisierungen nur gemeinsam mit ihren Nachbarn bewegt werden können. Ziel ist es nun, die D-dimensionale Darstellung der zu lernenden Mannigfaltigkeit mit dem Gitter wie mit einem elastischen Fischernetz abzudecken (siehe [84], S.136 f.).

Die m Prototypen ($m < n$), d.h. die Punkte des Gitters, haben sowohl d-dimensionale Darstellungen $g_r \in \mathbb{R}^d$ als auch D-dimensionale Darstellungen $c_r \in \mathbb{R}^D$ für $r = 1, \ldots, m$. Die Punkte g_r werden als d-dimensionales Gitter im Vorhinein festgelegt. In der Regel verteilt man die Punkte äquidistant. Weiterhin ist die Form der Nachbarschaften vorzugeben. Im Fall $d = 2$ wählt man üblicherweise quadratische (d.h. 8 Nachbarn) oder hexagonale (d.h. 6 Nachbarn) Nachbarschaften. In höheren Dimensionen sind Nachbarschaften in Form von Hyperwürfeln gebräuchlich. Damit ergibt sich für die globale Form des Gitters meist ein Rechteck oder Hexagon (für $d = 2$) bzw. allgemeiner ein Parallelepiped (für $d > 2$) (siehe [84], S.138). Die Punkte c_r sind zunächst unbekannt und werden durch die SOM bestimmt. Dies mag paradox erscheinen, da wir bei den bisher diskutierten Methoden immer die niedrigdimensionalen Punkte zu bestimmen hatten (vgl. [84], S.137). Die globale niedrigdimensionale Darstellung ist somit unabhängig von der Mannigfaltigkeit fixiert. Innerhalb dieser festen Form werden jedoch die Nachbarschaftsbeziehungen der hochdimensionalen Datenpunkte erhalten. Dies lässt sich schön nachvollziehen, wenn man die Datenpunkte einer gewissen „Region“ der zu lernenden Mannigfaltigkeit unterschiedlich färbt oder mit gewissen Labels versieht (vgl. [84], S.140).

Nach der Berechnung der $c_r \in \mathbb{R}^D$ für $r = 1, \ldots, m$ wählt man den niedrigdimensionalen Punkt y_i als den d-dimensionalen Gitterpunkt g_r, dessen zugehöriger D-dimensionaler Prototyp c_r am nächsten an x_i liegt, d.h.

$$y_i = g_r,$$

wobei

$$r = \underset{s=1,\dots,m}{\operatorname{argmin}} \|x_i - c_s\|_2.$$

Diese Berechnung der Punkte y_i ist natürlich recht grob, sodass relativ viele y_i identisch sein werden (vgl. [84], S.139). In [57] werden daher stattdessen interpolatorische Methoden für eine feinere Darstellung vorgeschlagen.

Die Bestimmung der Prototypen c_r erfolgt nach einer passenden Initialisierung iterativ in mehreren Schritten, sogenannten Epochen, wie im folgenden Algorithmus, der eine Art Robbins-Munro-Prozedur [113] darstellt, festgehalten ist.

Algorithmus 1.39 *Für jede Epoche:*
Für jeden Punkt x_i, d.h. für $i = 1, \dots, n$:

1. *Bestimme den Index r des zu x_i nächsten Prototypen, d.h.*

$$r = \underset{s=1,\dots,m}{\operatorname{argmin}} \|x_i - c_s\|_2.$$

2. *Aktualisiere alle Prototypen, indem man c_s durch*

$$c_s + \alpha \nu_\lambda(r, s)(x_i - c_s)$$

ersetzt. Dabei ist $\alpha \in [0, 1]$ die sogenannte Lernrate und ν_λ eine sogenannte Nachbarschaftsfunktion.

Bemerkung 1.40

1. *Es werden soviele Epochen durchlaufen, bis Konvergenz (im Sinne hinreichend kleiner Aktualisierungen der Prototypen) eintritt.*

2. *Für die Nachbarschaftsfunktion ν_λ wird z.B.*

$$\nu_\lambda(r, s) = \begin{cases} 0, & \text{falls } \|g_r - g_s\| > \lambda \\ 1, & \text{falls } \|g_r - g_s\| \leq \lambda \end{cases}$$

gewählt. Dabei wählt man den Parameter λ wie auch die Lernrate α monoton fallend von Epoche zu Epoche (vgl. [84], S.138).

3. *Man beachte, dass die Aktualisierungsvorschrift empirisch induziert ist. Es ist im Allgemeinen keine explizite Zielfunktion bekannt, welche die SOM optimieren (siehe [84], S.139).*

4. *Für die arithmetische Komplexität der SOM ergibt sich eine Größenordnung von $\mathcal{O}(nmD)$ pro Epoche des Lernschemas (siehe [84], S.139), wobei m die Anzahl der verwendeten Prototypen ist.*

Die SOM stellen einen relativ einfachen, robusten Algorithmus dar und gelten als Referenzmethode für die 2D-Visualisierung von Daten (siehe [84], S.141). Dass die Form der niedrigdimensionalen Einbettung als das Gitter quasi vorgegeben wird, ist jedoch nicht für jede Anwendung sinnvoll. Topologie erhaltende Methoden wie die LLE, die Laplaceschen Eigenabbildungen (beide in Unterabschnitt 1.2.2 vorgestellt) oder Isotop (folgt in diesem Unterabschnitt), welche ein dateninduziertes Gitter (statt eines vorgegebenen) nutzen, können hier im Vorteil sein (vgl. [84], S.187). Weiterhin kann problematisch sein, dass sich viele Implementierungen der SOM auf den Fall $d = 1$ bzw. $d = 2$ beschränken (siehe [84], S.141).

Die durch Anwendung der SOM beispielhaft gelieferte niedrigdimensionale Darstellung einer Schweizer Rolle findet man etwa in [84](S.140). Es sei nochmals darauf hingewiesen, dass sich das Vorgehen der SOM stark von den anderen in dieser Arbeit diskutierten Methoden unterscheidet. Nach Konstruktion entspricht die niedrigdimensionale Darstellung unabhängig von den vorliegenden hochdimensionalen Daten dem vordefinierten Gitter. Die Aussagekraft der Darstellung liegt in der Platzierung der den Punkten zugeordneten Labels (vgl. [84], S.140).

Bemerkung 1.41 *1. Es existieren zahlreiche Modifikationen der SOM, darunter z.B. die wachsende Zellstruktur (engl. Growing Cell Structure, GCS) [51] und das wachsende Gitter (engl. Growing Grid, GG) [52] sowie die wachsenden SOM (engl. Growing SOM, GSOM) [11]. Bei diesen Varianten wächst die Anzahl der Prototypen automatisch mit der Komplexität des Problems (vgl. [84], S.142).*

2. Gewissermaßen stellt die von Bishop et al. entwickelte generative topographische Abbildung (engl. Generative Topographic Mapping, GTM) [19, 20, 125], welche auf dem Konzept Bayesschen Lernens und dem EM-Algorithmus (engl. expectation maximization algorithm) basiert, eine probabilistische Erweiterung der SOM dar (vgl. [84], S.142 f.).

Isotop

Isotop - von Lee et al. in [81] vorgeschlagen - ist eine Methode, die ähnlich zu den SOM ist, aber die Vektorquantisierung und Dimensionsreduktion trennt. Dadurch ergibt sich bei der Form der niedrigdimensionalen Darstellung im Gegensatz zu den SOM, wo die Darstellung im Vorhinein durch ein Gitter definiert wird, mehr Freiheit (vgl. [84], S.165). Isotop besteht aus drei separaten Schritten:

1. Vektorquantisierung (optional)

2. Konstruktion eines Nachbarschaftsgraphen

3. Berechnung der niedrigdimensionalen Darstellung .

Schritt 1, die Vektorquantisierung, ist optional und muss nur durchgeführt werden, wenn der Datensatz sehr groß ist.

In Schritt 2 wird der Datensatz in einen gewichteten Graphen übersetzt, der die Topologie der zugrundeliegenden Mannigfaltigkeit erfassen soll. Die Knoten des Graphen sind die gegebenen Datenpunkte x_i. Zwei Knoten werden mit einer Kante verbunden, wenn die zugehörigen Datenpunkte Nachbarn sind, und die Kante wird mit dem euklidischen Abstand der betreffenden Punkte gewichtet. Wie bei der Methode Isomap berechnet man nun für je zwei Punkte x_i und x_j mittels des Algorithmus von Dijkstra [45] die Graph-Distanz $d_G(i,j)$, d.h. die Länge des kürzesten Pfades zwischen den entsprechenden Knoten im Graphen G, als Approximation der geodätischen Distanz zwischen x_i und x_j.

Die Berechnung der niedrigdimensionalen Darstellung in Schritt 3 von Isotop entspricht nun einer Rücktransformation der Informationen des Graphen in eine d-dimensionale Punktkonfiguration (siehe [84], S.166). Dabei ersetzt man die Knoten des Graphen bzw. die D-dimensionalen Datenpunkte x_i durch d-dimensionale Punkte y_i. Alle Punkte y_i werden zunächst als Nullvektor initialisiert und in der Folge durch eine ähnliche Lernprozedur wie bei den SOM iterativ aktualisiert. Dazu definiert man d-dimensionale Normalverteilungen $\mathcal{N}(y_i, I)$ mit Zentrum in y_i und der Einheitsmatrix $I \in \mathbb{R}^{d\times d}$ als Kovarianzmatrix. Man verwendet nun den folgenden Algorithmus (vgl. [84], S.167), wobei m die Anzahl der Prototypen bei angewandter Vektorquantisierung sei.

Algorithmus 1.42 *Für jeden Punkt $y_i,\ i = 1,\dots,m$:*

1. *Ziehe zufällig einen Punkt z aus der Normalverteilung $\mathcal{N}(y_i, I)$.*

2. *Bestimme den Index j_0 des zu z nächsten Punktes y_{j_0}:*

$$j_0 = \operatorname*{argmin}_{l=1,\dots,m} \|z - y_l\|_2.$$

3. *Aktualisiere alle Punkte $y_l,\ l \in \{1,\dots,m\} \setminus \{j_0\}$ durch*

$$y_l + \alpha\nu_\lambda(l, j_0)(z - y_l)$$

mit einer Lernrate $0 \le \alpha \le 1$ und einer Nachbarschaftsfunktion ν_λ mit Nachbarschaftsweite $\lambda > 0$.

Bemerkung 1.43 *1. Wie bei den SOM und der CCA nennt man einen Durchlauf des Aktualisierungsalgorithmus Epoche. Man durchläuft so viele Epochen, bis Konvergenz (im Sinne hinreichend kleiner Aktualisierungen der Punkte y_i) eintritt.*

2. *Für die Nachbarschaftsfunktion ν_λ wählt man etwa*

$$\nu_\lambda(i,j) = \exp\left(-\frac{1}{2}\frac{d_G^2(i,j)}{\lambda^2 D_G^2(j)}\right)$$

mit

$$D_G(j) = \frac{1}{|\{h : (h,j) \in G\}|} \sum_{\{h:(h,j)\in G\}} d_G(h,j).$$

Dabei bedeutet $(h,j) \in G$, dass die Knoten mit Index h und j im Nachbarschaftsgraphen G durch eine Kante verbunden sind. Der Term im Argument der Nachbarschaftsfunktion ist somit eine Art relative Distanz der Knoten mit Index i und j, welche verhindert, dass in dichter abgetasteten Teilen der zugrundeliegenden Mannigfaltigkeit kleinere Abstände gemessen werden (siehe [84], S.167).

3. *Sinn des Aktualisierungsschrittes mittels der Nachbarschaftsfunktion ν_λ ist es, die Topologie der Mannigfaltigkeit in Form der paarweisen Nachbarschaften in der niedrigdimensionalen Darstellung zu erhalten (vgl. [84], S.168).*

4. *Wie bei den SOM wurde die Aktualisierungsvorschrift empirisch bedingt aufgestellt und eine konkrete Zielfunktion, welche Isotop optimiert, ist bisher unbekannt (siehe [84], S.171).*

5. *Ist m die Anzahl der betrachteten Prototypen, so erfordert die Berechnung der Graph-Distanzen $\mathcal{O}(m^2 \log m)$ Operationen. Die arithmetische Komplexität der Lernphase ist von der Ordnung $\mathcal{O}(m^2)$ pro Epoche (siehe [84], S.168).*

Für ein Beispiel einer niedrigdimensionalen Darstellung einer Schweizer Rolle sei auf [84](S.170) verwiesen. Auch wenn die Darstellung kleinere „Wellen“ aufweist, erkennt Isotop in diesem Fall sehr gut die zugrundeliegende Struktur.

Bemerkung 1.44 *Die Dimensionsreduktionsmethode der stochastischen Nachbarschaftseinbettung (engl. Stochastic Neighbor Embedding, SNE) [70] kann als eine probabilistische Variante von Isotop verstanden werden (siehe [84], S.171). Die SNE setzt im hoch- und im niedrigdimensionalen Raum Gauß-Verteilungen für die (bedingte) Wahrscheinlichkeit, dass Punkte Nachbarn sind, an und ermittelt die niedrigdimensionale Darstellung derart, dass die beiden Wahrscheinlichkeitsverteilungen bestmöglich zueinanderpassen. Hierzu minimiert man ein Kostenfunktional, welches aus einer Summe von Kullback-Leibler-Divergenzen besteht, mittels eines Gradientenverfahrens. Eine Weiterentwicklung der SNE wurde in Form der stochastischen Nachbarschaftseinbettung mit Student-t-Verteilung (engl. t-Distributed Stochastic Neighbor Embedding, t-SNE) [90] vorgeschlagen. Gegenüber der ursprünglichen SNE werden bei der t-SNE symmetrisierte Wahrscheinlichkeiten sowie im niedrigdimensionalen Raum eine Student-t-Verteilung mit einem Freiheitsgrad statt der Gauß-Verteilung benutzt.*

1.2.5 Sonstige Methoden

In diesem Unterabschnitt betrachten wir noch zwei Dimensionsreduktionsmethoden, welche sich nicht in die Gliederung der restlichen Methoden einfügen lassen. Die Methode der lokal linearen Koordination (LLC) weist einen Hybridcharakter auf und verbindet lokale und globale Dimensionsreduktion. Die Methode der mehrschichtigen Autoencoder ist der Welt der künstlichen neuronalen Netze zuzuordnen, nutzt jedoch keine direkte geometrische Motivation wie die CCA, die SOM oder Isotop.

Lokal lineare Koordination (LLC)

Die lokal lineare Koordination (engl. Locally Linear Coordination, LLC) [127] ordnet unterschiedliche lokal lineare Modelle global zu einer niedrigdimensionalen Darstellung der gegeben Daten an (vgl. [91]).

Zunächst bestimmt die LLC eine Mischung von m lokal linearen Modellen gemäß einer Faktor-Analyse [123] (engl. mixture of factor analyzers) mittels des EM-Algorithmus (engl. expectation maximization algorithm). Anstelle der Faktor-Analyse lassen sich auch eine probabilistische PCA (engl. mixture of probabilistic PCA) (siehe [131]) oder für mehr Robustheit gegenüber Ausreißern Student-t-verteilte Unterraum-Modelle (engl. mixture of t-distributed subspaces) (siehe [112]) verwenden. Grundsätzlich ist jede Mischung von lokalen Dimensionsreduktionsmethoden denkbar (siehe [127]).

Man erhält zu jedem Datenpunkt x_i und jedem $j \in \{1, \ldots, m\}$ Darstellungen z_{ij} und zugehörige Gewichte (engl. responsibilities) r_{ij} mit $r_{ij} \geq 0$ und $\sum_{j=1}^{m} r_{ij} = 1$, welche quantifizieren, inwieweit das m-te Modell x_i darstellt. Man setze

$$u_{ij} = r_{ij} z_{ij}$$

und speichere diese gewichteten Darstellungen u_{ij} in einer $n \times mD$-Blockmatrix U.

In einem zweiten Schritt werden die lokalen Modelle nun mittels der konvexen Kostenfunktion der LLE koordiniert. Man bestimmt wie bei der LLE für jeden Datenpunkt x_i Rekonstruktionsgewichte, die eine optimale Affinkombination bezüglich seiner Nachbarn bilden, und speichert diese in der Matrix W. Anschließend setzt man

$$A = U^T (I - W)^T (I - W) U,$$
$$B = \frac{1}{n} U^T U.$$

Mittels der zu den d kleinsten von Null verschiedenen Eigenwerten zugehörigen Eigenvektoren des verallgemeinerten Eigenwertproblems

$$Av = \lambda B v$$

definiert man eine Abbildungsmatrix $L = (v_1, \ldots, v_d)$. Die lineare Abbildung L wird schließlich auf die Matrix U der gewichteten (lokalen) Darstellungen angewendet, um die (globale) niedrigdimensionale Darstellung der Daten zu erhalten, d.h.

$$Y = UL.$$

Bemerkung 1.45 *1. Man beachte, dass die Abbildung der gewichteten lokalen Darstellungen u_{ij} auf y_i somit linear, die Abbildung der hochdimensionalen Datenpunkte x_i auf y_i durch die Kombination vieler lokaler Modelle jedoch nicht linear ist (vgl. [127]).*

2. Obwohl die LLC in ihrem zweiten Schritt ein konvexes Problem mittels Spektraltechnik löst, ist sie aufgrund ihres ersten Schrittes als nicht konvexe Methode zu klassifizieren. Der im ersten Schritt durchgeführte EM-Algorithmus zur Berechnung der lokalen Modelle erweist sich als kritischste Stelle der LLC, da er in einem lokalen Maximum stoppen kann (siehe [91]).

3. Die totale arithmetische Komplexität der LLC ergibt sich zu einer Größe von $\mathcal{O}(Tmd^3)$ und der Speicherplatzbedarf zu einer Größe von $\mathcal{O}(nmd)$ (siehe [91]). Dabei ist m die Anzahl der betrachteten lokalen Modelle und T die Anzahl der Iterationen des EM-Algorithmus.

4. Da die LLC eine konkrete Abbildungsvorschrift der hoch- auf die niedrigdimensionalen Datenpunkte bestimmt, ist eine Verallgemeinerung der niedrigdimensionalen Darstellung auf zusätzliche Punkte (Out-of-sample-Erweiterung) problemlos möglich (vgl. [91]).

In Abbildung 1.3 findet sich eine gemäß LLC berechnete niedrigdimensionalen Darstellung der Schweizer Rolle aus Abbildung 1.2. Die Darstellung weiß in diesem Beispiel überhaupt nicht zu überzeugen. Es wurden $m = 20$ lokale Modelle betrachtet und im LLE-Schritt wurde der Nachbarschaftsparameter $k = 12$ gewählt.

Bemerkung 1.46 *Eine der LLC sehr ähnliche Dimensionsreduktionsmethode stellt die Mannigfaltigkeitskartierung (engl. Manifold Charting) [22] dar. Wie bei der LLC erfolgt bei der Mannigfaltigkeitskartierung eine globale Koordination lokaler Modelle, welche aus einer Faktor-Analyse oder einer probabilistischen PCA resultieren. Die Bestimmung der linearen Abbildung von den gewichteten lokalen Darstellungen auf die globalen Koordinaten erfolgt jedoch durch Minimierung eines anderen Kostenfunktionals, wobei die optimale Lösung wieder durch Spektraltechniken gefunden werden kann (siehe [91]).*

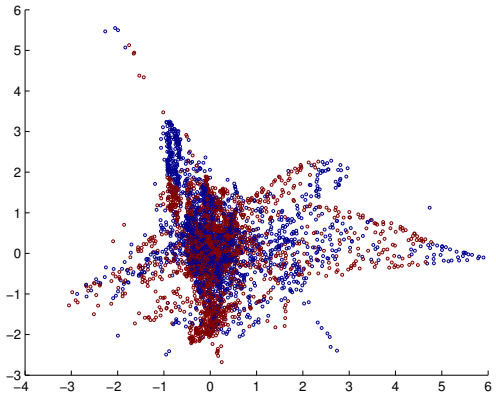

Abbildung 1.13: Mittels der Toolbox [89] erhaltene zweidimensionale Darstellung der Schweizer Rolle gemäß LLC.

Mehrschichtige Autoencoder (Multilayer Autoencoders)

Die Methode der mehrschichtigen Autoencoder (engl. Multilayer Autoencoder) [44, 71] entstammt dem Gebiet künstlicher neuronaler Netze (engl. artificial neural networks) und lässt sich für eine Form nicht geometrisch motivierter Dimensionsreduktion nutzen.

Wir gehen von einem künstlichen neuronalen Netz mit einer Eingabeschicht (engl. input layer) und einer Ausgabeschicht (engl. output layer) mit jeweils D Knoten, den künstliche Neuronen, aus. Zwischen Eingabe- und Ausgabeschicht befindet sich eine ungerade Anzahl von verdeckten Schichten (engl. hidden layers). Die mittlere verdeckte Schicht hat d Knoten (siehe [91]).

Die Knoten einer Schicht sind über gewichtete Kanten mit den Knoten der folgenden Schicht verbunden. Wir gehen von vorwärts gespeisten neuronalen Netzen (engl. feed-forward neural networks) aus, d.h., dass Kanten nur vorwärts gerichtet sind. Der einem Knoten zugewiesene Wert ergibt sich im Wesentlichen durch Anwendung einer Aktivierungsfunktion auf die gewichtete Summe der Werte der Knoten in der vorherigen Schicht. Für die Aktivierungsfunktionen wählt man mit Ausnahme der mittleren verdeckten Schicht, wo man lineare Aktivierungsfunktionen verwendet, üblicherweise Sigmoidfunktionen, um nicht lineare Zusammenhänge erfassen zu können (vgl. [91]). Eine Sigmoidfunktion ist dabei eine reellwertige, beschränkte und differenzierbare Funktion mit überall positiver (oder überall negativer) erster Ableitung (siehe etwa [64]), z.B. die sogenannte logistische Funktion $\phi : \mathbb{R} \to \mathbb{R},\ \phi(t) = \frac{1}{1+e^{-t}}$. Ein neuronales Netz dieser Form wird auch mehrschichtiges Perzeptron (engl. multilayer perceptron, MLP) genannt. Für eine genauere Beschreibung künstlicher neuronaler Netze sei etwa auf [18] verwiesen.

Das Voranschreiten im neuronalen Netz von der Eingabeschicht bis zur mittleren verdeckten Schicht entspricht einer „Kodierung“ der Eingabedaten (Encoder), das Voranschreiten von der mittleren verdeckten Schicht zur Ausgabeschicht einer Rekonstruktion der Eingabedaten aus der gefundenen „Kodierung“ (Decoder) (vgl. [71]). Man setzt für die Knoten der Eingabeschicht die gegebenen hochdimensionalen Datenpunkte x_i an und trainiert das neuronale Netz so, dass der mittlere quadratische Fehler zwischen Eingabe und Ausgabe möglichst gering wird. Die Werte der d Knoten der mittleren verdeckten Schicht lassen sich dann als die gesuchten niedrigdimensionalen Datenpunkte y_i auffassen (siehe [91]).

Das Training eines neuronalen Netzwerkes erfolgt üblicherweise über Rückpropagierung (engl. backpropagation). Diese Ansätze konvergieren jedoch nur langsam und sind anfällig dafür, in lokalen Minima zu stoppen (siehe [91]). In [69] wird stattdessen ein dreischrittiger Lernprozess vorgeschlagen. Zunächst werden die Encoder-Schichten nacheinander durch das Trainieren sogenannter eingeschränkter Boltzmann-Maschinen (engl. Restricted Boltzmann Machine, RBM) mittels einer in [68] beschriebenen Lernprozedur eingelernt. Zweitens erhält man durch Invertierung der Encoder-Schichten die Decoder-Schichten. Im dritten Schritt erfolgt eine Feinabstimmung der Gewichte durch einen Rückpropagierungsansatz (vgl. [91]).

Bemerkung 1.47 *1. Die Größenordnung der arithmetischen Komplexität der gesamten Methode beträgt $\mathcal{O}(Tnw)$, wobei T die Anzahl der Iterationen des Optimierungsalgorithmus und w die Anzahl der Gewichte darstellt. Der Speicherplatzdarf liegt in der Ordnung $\mathcal{O}(w)$ (siehe [91]).*

2. Man beachte, dass bei der Methode der mehrschichtigen Autoencoder ein parametrischer Zusammenhang zwischen hoch- und niedrigdimensionalen Datenpunkten konstruiert wird. Dadurch erhält man ohne weiteren Aufwand eine Verallgemeinerung der niedrigdimensionalen Darstellung auf zusätzliche Datenpunkte (Out-of-sample-Erweiterung) (siehe [91]).

Vorteilhaft ist die tiefgehende Struktur neuronaler Netze. Die Methode der mehrschichtigen Autoencoder konstruiert die niedrigdimensionalen Darstellungen y_i aus den gegebenen hochdimensionalen Punkten x_i über mehrere Zwischenschritte, die Schichten des neuronalen Netzes, und berücksichtigt somit mehrfach nicht lineare Zusammenhänge. Die zuvor vorgestellten Spektralmethoden sind dagegen gewissermaßen einschichtige Methoden, da sie nur einen einfachen nicht linearen Zusammenhang zwischen den hoch- und niedrigdimensionalen Punkten betrachten (vgl. [91]).

Die Methode der mehrschichtigen Autoencoder ist nicht praktikabel für Datensätze zu hoher Dimension, da man in diesem Fall eine zu hohe Anzahl von Gewichten benötigt. Eine Vorbehandlung der Daten mittels einer PCA, um die Dimension des Datensatzes

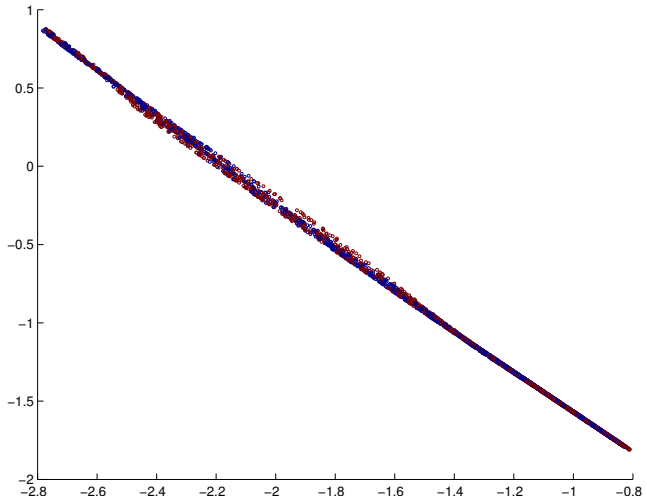

Abbildung 1.14: Mittels der Toolbox [89] erhaltene zweidimensionale Darstellung der Schweizer Rolle gemäß mehrschichtiger Autoencoder.

vorab zu verringern, könnte hier als Gegenmittel dienen (vgl. [91]). Man beachte auch, dass eine ausreichende Menge von Datenpunkten erforderlich ist, damit das Training des neuronalen Netzwerkes zufriedenstellende Ergebnisse liefert (siehe [91]).

Abbildung 1.3 illustriert eine enttäuschende Leistung mehrschichtiger Autoencoder bei der niedrigdimensionalen Darstellung der Schweizer Rolle aus Abbildung 1.2. Die Daten kollabieren in diesem Beispiel auf eine Gerade. Es geht somit eine Dimension des Datensatzes verloren. Es sei jedoch auf überzeugende Ergebnisse der mehrschichtigen Autoencoder für einige natürliche Datensätze hingewiesen (siehe [91]).

1.2.6 Fazit

Wir schließen dieses Kapitel mit einem Fazit, welches globale und lokale Spektralmethoden bzw. Spektral- und Nicht-Spektralmethoden gegenüberstellt. Wir orientieren uns dabei insbesondere an den Ausführungen in [91] und [84](S.238 ff.).

Globale gegen lokale Spektralmethoden

Aufgrund ihrer Konstruktion erhalten lokale Spektralmethoden im Gegensatz zu den globalen Spektralmethoden lokale Eigenschaften einer Mannigfaltigkeit besser. Nachbarschaften aus den hochdimensionalen Daten werden in die niedrigdimensionale Darstellung übernommen. Dieser Vorteil zeigt sich insbesondere bei stark gekrümmten Man-

nigfaltigkeiten (vgl. [28]). Die Trenunng zwischen „global“ und „lokal“ entspricht hier auch einer Abgrenzung von abstandserhaltenden und Topologie erhaltenden Spektralmethoden des Manifold-Learning.

Weiterhin ist zu beachten, dass bei lokalen Spektralmethoden die Spektralzerlegung einer dünnbesetzten und bei globalen Spektralmethoden die einer vollbesetzten Matrix erfolgt (vgl. [28, 91]). Durch die Verwendung von Arnoldi- oder Jacobi-Davidson-Methoden [6, 49] lässt sich die Spektralzerlegung für dünnbesetze Matrizen wesentlich beschleunigen.

Dennoch enttäuschen lokale Methoden bei der Dimensionsreduktion vieler Datensätze aus praktischen Anwendungen (siehe [91]). Grund dafür sind einige der konzeptionellen bzw. praktischen Nachteile der lokalen Methoden.

Die LLE, die Laplaceschen Eigenabbildungen und die HLLE leiden unter der Existenz trivialer Lösungen Y des jeweiligen Optimierungsproblems. Die aufgrund dessen eingeführten Nebenbedingungen für die Kovarianzmatrix von Y sind nicht immer ausreichend, um die ungewünschten trivialen Lösungen zu unterdrücken. Es besteht die Tendenz, Lösungen zu produzieren, die große Teile der Daten auf den Ursprung abbilden und einzelne austretende Strahlen zur Erfüllung der Nebenbedingungen aufweisen (siehe [91]). Auch können die Nebenbedingungen zu unerwünschten Skalierungen des Datensatzes führen (siehe [55]).

Mannigfaltigkeiten mit Unstetigkeiten oder nicht zusammenhängende Mannigfaltigkeiten können den lokalen Methoden Probleme bereiten, weil diese Phänomene nicht mit der impliziten Annahme lokaler Linearität zu vereinbaren sind. Dies ist vor allem in der praktischen Anwendung kritisch, da natürliche Datensätze oft unstetig oder nicht zusammenhängend sind (siehe [91]).

Alle lokalen Spektralmethoden benötigen die Bestimmung von Nachbarschaften von Datenpunkten. Diese kann Schwierigkeiten bereiten, wenn die Abtastung der Mannigfaltigkeit nicht dicht genug ist, was vor allem in sehr hoher Dimension kritisch ist (Fluch der Dimension). Die Problematik, den Nachbarschaftsparameter passend zu wählen, ist besonders relevant, wenn Rauschen oder Ausreißer im Datensatz vorliegen. Daraus resultierende möglicherweise unpassende Nachbarschaftsbeziehungen können Erkenntnisse bezüglich der globalen Struktur der Mannigfaltigkeit verfälschen (vgl. [91]).

Ein weiteres Problem tritt bei der Lösung der Eigenwertprobleme auf. Das Spektrum der betrachteten Matrizen ist üblicherweise sehr breit. Die kleinsten Eigenwerte können etwa von der Größenordnung 10^{-7} sein und die größten Eigenwerten von der Größenordnung 10^2 (siehe [91]). Das Lösen derartiger Eigenwertprobleme ist instabil. Die lokalen Methoden benötigen die d kleinsten Eigenwerte des jeweiligen Eigenwertproblems, welche unter Umständen nicht korrekt erkannt werden, da sie nicht von den trivialen Null-Eigenwerten zu unterscheiden sind. Die globalen Spektralmethoden verwenden hingegen

die größten Eigenwerte der betreffenden Eigenwertprobleme und sind somit weniger von den Instabilitäten betroffen (vgl. [91]).

Spektralmethoden gegen Nicht-Spektralmethoden

Spektralmethoden zur Dimensionsreduktion sind gleichermaßen einfach und theoretisch fundiert (siehe [84], S.240). Die zugehörigen Optimierungsprobleme lassen sich, auch wenn die im vorangegangenen Unterabschnitt dieses Fazits angesprochenen Probleme zu beachten sind, exakt mittels der üblichen effizienten Algorithmen für Eigenwertprobleme lösen (vgl. [91]). Die Optimierungsprobleme der Nicht-Spektralmethoden sind aufwändiger zu lösen. Hierzu sind iterative Techniken wie Quasi-Newton-Verfahren oder stochastische Gradientenverfahren notwendig, welche unter Umständen langsam oder nur gegen lokale Extrema konvergieren (vgl. [91]). Bei den SOM und Isotop ist die zu optimierende Zielfunktion nicht einmal bekannt (siehe [84], S.141, S.171).

Nicht-Spektraltechniken bieten jedoch mehr Flexibilität als Spektraltechniken (siehe [84], S.241). Spektraltechniken gehen zur Dimensionsreduktion gewissermaßen in zwei Schritten vor. Im nicht linearen ersten Schritt wird der Datensatz implizit in einen hochdimensionalen Merkmalsraum abgebildet (vgl. die Ausführungen zur Methode KPCA). Die eigentliche Dimensionsreduktion erfolgt in einem zweiten linearen Schritt, z.B. in Form einer Abstandserhaltung. Die Abbildung im zweiten Schritt wird optimiert ausgewählt. Die Auswahl des Kerns im ersten Schritt wird hingegen nicht optimiert und ist somit ein Stück weit willkürlich (siehe [84], S.239). Eine Ausnahme bildet die MVU, wo auch im ersten Schritt eine optimierte Auswahl des Kerns erfolgt, was jedoch mit einem enormen rechnerischen Aufwand bezahlt wird. Bei den Nicht-Spektralmethoden gibt der Verzicht auf Konvexität der Zielfunktion mehr Spielraum, die typischen Nachteile der konvexen Zielfunktionen (siehe vorangegangenen Unterabschnitt dieses Fazits) zu vermeiden (siehe [91]).

Ein Vorteil der Spektralmethoden ist, dass sie sogenannte inkrementelle niedrigdimensionale Darstellungen zulassen. Führt man die jeweilige Spektralzerlegung vollständig durch, so erhält man niedrigdimensionale Darstellungen für alle Zieldimension $d = 1, \ldots, D$, indem entsprechend viele Eigenwerte und Eigenvektoren berücksichtigt bzw. nicht berücksichtigt werden. Dies ist besonders praktisch, wenn man sich vorab unsicher hinsichtlich der korrekten Zieldimension d ist. Inkrementelle Einbettungen sind bei den Nicht-Spektralmethoden nicht möglich. Hier muss für jede vorgegebene Zieldimension eine niedrigdimensionale Darstellung von Grund auf neu berechnet werden. Dies kann jedoch auch als Vorteil gesehen werden, da die niedrigdimensionale Darstellung für die betreffende Zieldimension spezifischer bestimmt wird (vgl. [84], S.42).

2 Wavelets entlang von Pfaden zur Entstörung gestreuter Daten

2.1 Einleitung und verwandte Arbeiten

Entstörung ist ein wesentlicher Bestandteil der Verarbeitung von Daten. In vielen Anwendungen sind möglichst störungsfreie Daten Voraussetzung für eine erfolgreiche Weiterverarbeitung, z.B. mit Dimensionsreduktionsmethoden wie in Kapitel 1 beschrieben. Wavelet-Shrinkage-Methoden haben sich zur Entstörung von Signalen oder Bildern, d.h. im ein- oder zweidimensionalen Fall, bewährt [31]. Mittels der Wavelet-Transformation erfolgt eine Trennung des Signals bzw. Bildes in hohe und tiefe Frequenzen. Störungen werden als hochfrequent angenommen und folglich durch ein Shrinkage (Thresholding) der entsprechenden Hochpass-Koeffizienten eliminiert. Im Falle eines Bildes liegen die Daten auf einem äquidistanten Gitter, den Bildpixeln, vor. Für nicht äquidistante Datenpunkte ist jedoch die übliche Tensorprodukt-Wavelet-Transformation nicht anwendbar.

Es existieren diverse Wavelet-Konstruktionen für den nicht äquidistanten Fall. Dabei erfolgt entweder wie in [5, 25, 61, 76] zunächst eine Approximation mit dem Ziel, auf den äquidistanten Fall zurückzukommen, oder die Konstruktion beruht auf Wavelets zweiter Generation (engl. second generation wavelets) [126] mittels des Lifting-Schemas [10, 40, 73, 136]. Alle diese Methoden büßen jedoch einen Teil der Effizienz und Einfachheit der üblichen Wavelet-Transformation ein.

In diesem Kapitel wird eine von uns in [66] neu eingeführte adaptive Wavelet-Shrinkage-Methode zur Entstörung von Funktionswerten, welche auf hochdimensionalen gestreuten Datenpunkten gegeben sind, diskutiert und weiterentwickelt. Die Methode stellt eine Verallgemeinerung der EPWT („Easy Path Wavelet Transform“) dar. Eine Zusammenfassung des Rahmenkonzeptes der EPWT und der neuen Entstörungsmethode findet sich in [58]. Die EPWT wurde von Plonka in [101] zur Bildkompression vorgeschlagen und lässt sich - ähnlich wie in [103, 104] für Daten auf der Sphäre geschehen - auf den Fall gestreuter Punkte $x_j \in \mathbb{R}^d$ mit zugeordneten Funktionswerten $f(x_j) \in \mathbb{R}$ übertragen. Grundlage ist eine eindimensionale Wavelet-Transformation entlang von sinnvoll zu konstruierenden Pfaden. In Anwesenheit von Rauschen müssen wir jedoch die Pfadkonstruktion der EPWT modifizieren. Weiterhin verwenden wir eine Durchschnittsbildung über

mehrere mittels unterschiedlicher Pfade erhaltene entstörte Versionen des Datensatzes, um das Entstörungsergebnis durch Ausnutzung von Redundanz zu verbessern. Dieses Vorgehen ist angelehnt an die von Coifman und Donoho vorgeschlagene translationsinvariante Entstörung (engl. translation-invariant de-noising) [31], auch Cycle-Spinning genannt.

Die EPWT teilt gewissermaßen die Grundidee der geometrischen Grouplets [94], bei denen nach einer Gruppierung der vorliegenden Punkte durch sogenannte Assoziationsfelder (engl. association fields) eine gewichtete Haar-Wavelet-Transformation durchgeführt wird. Weiterhin zeigt die GTBWT (Generalized Tree-based Wavelet Transform) von Ram et al. [108], eine Wavelet-Konstruktion für gestreute Daten oder Daten auf Graphen, Parallelen zur EPWT bzw. zur neu eingeführte Entstörungsmethode. Die dort verwendeten Bäume (engl. trees) entsprechen dem Konzept der Pfade des im Folgenden in dieser Arbeit diskutierten Entstörungsansatzes. Die GTBWT stellt eine Verallgemeinerung der Haar-ähnlichen Wavelet-Transformation in [53] dar. Ram et al. schlagen zudem in [109] mit der RTBWT (Redundant Tree-based Wavelet Transform) eine Modifikation der GTBWT vor, welche auf einer redundanten Wavelet-Filterbank beruht. Des Weiteren ist hier das Schema derselben Autoren aus [110] für Zwecke der Bildentstörung oder des Inpaintings zu nennen, welches auf einer Umordnung der Blöcke des Bildes und anschließender Anwendung spezieller Filter basiert. Letztgenanntes Schema weist wiederum auch größere Ähnlichkeiten zum von Dabov et al. eingeführten BM3D-Algorithmus [36] zur Bildentstörung auf, dessen Grundidee in Abschnitt 2.7 kurz diskutiert wird. Schließlich besteht eine lose Verwandtschaft der in [38] von Dekel und Leviatan und in [39] von Dekel und Nemirovksy diskutierten sogenannten geometrischen Wavelets (engl. geometric wavelets) zu unserem Entstörungsalgorithmus. Bei den geometrischen Wavelets handelt es sich um ein spezielles adaptives Wavelet-Konzept, welches auf einer binären Raumpartitionierung (engl. binary space partition) beruht und in [39] mit der Idee sogenannter „Zufallswälder“ (engl. random forests) verbunden wird.

Eine weitere spezielle Multiskalenanalyse auf Mannigfaltigkeiten und Graphen konstruieren Coifman und Maggioni in [34] in Form der sogenannten Diffusionswavelets (engl. diffusion wavelets). Wesentlich ist hier die Verwendung gewisser Diffusionsoperatoren für die Konstruktion orthogonaler Basen statt des Dilatationsoperators bei klassischen Wavelets. Die Diffusionswavelets sind demselben Rahmenkonzept wie die Diffusionsabbildungen und Laplaceschen Eigenabbildungen (siehe auch Kapitel 1) zuzuordnen. In ähnlicher Weise werden in [63] Wavelets auf gewichteten Graphen konstruiert. Die Skalierung erfolgt dabei im Spektralbereich des Graph-Laplace-Operators des vorliegenden Graphen. In diesem Zusammenhang ist auch das Vorgehen der Autoren in [48] zur Approximation von Funktionswerten auf hochdimensionalen unstrukturierten Daten zu nennen. Hierbei erfolgt gewissermaßen eine Verknüpfung von Manifold-Learning und Approximation unter Zuhilfenahme der sogenannten Diffusionspolynome (engl. diffusion polynomials).

Für den Spezialfall der Entstörung von Bildern sind weitere auf der Konstruktion und Manipulation bildspezifischer Graphen beruhende Entstörungsmethoden wie z.B. [7, 97] bekannt, die jedoch nicht mit Wavelets arbeiten. Die Knoten dieser Graphen stellen die Bildpixel dar, wobei benachbarte Pixel mit Kanten verbunden werden. Diese Graphen zeigen somit Berührungspunkte zu den Pfadkonstruktionen, welche bei der in dieser Arbeit vorgestellten Entstörungsmethode benutzt werden. In [7] wird die Konstruktion spezieller sogenannter charakteristischer Graphen mit einem Regularisierungsansatz wie im klassischen ROF-Modell [115] verknüpft. Die Autoren von [97] verwenden eine sogenannte Gitterglättung (engl. grid smoothing) bei der die Gitterpunkte, d.h. die Pixel, unter der Annahme, dass Bildbereiche mit kleiner Varianz weniger repräsentierende Gitterpunkte benötigen, modifiziert werden.

Es sei weiter auf Ansätze wie [65, 56, 138, 134] zur Entstörung von Mannigfaltigkeiten hingewiesen. Im Unterschied zur in dieser Arbeit thematisierten Problemstellung sind in diesen Arbeiten keine Funktionswerte auf den hochdimensionalen gestreuten Daten gegeben. Stattdessen sollen hier Störungen der hochdimensionalen gestreuten Datenpunkte selbst beseitigt werden. Die hierzu vorgeschlagenen Techniken sind oft an Methoden des Manifold-Learning, wie in Kapitel 1 beschrieben, angelehnt. In [65] werden Diffusionsmethoden mit dem Graph-Laplace-Operator betrachtet. Die Autoren von [56] bzw. [138] nutzen lokal lineare Entstörung (engl. Locally Linear Denoising, LLD) bzw. Sparse-Subspace-Denoising genannte Methoden, welche auf Ideen der Hauptkomponentenanalyse, lokal linearer Einbettung (LLE) und lokal linearer Koordination (LLC) beruhen. Der Ansatz in [134] stützt sich wie einige Methoden des Manifold-Learning auf die Annahme, dass lokale Nachbarschaften von Punkten der Mannigfaltigkeit zu linearen Unterräumen gehören, und greift auf eine Maximum-Likelihood-Methode zurück.

Ausblick Im Folgenden werden wir in Abschnitt 2.2 zunächst eine allgemeine Einführung in Wavelet-Filterbänke liefern. Anschließend beschreiben wir in Abschnitt 2.3 den vorgeschlagenen Entstörungsalgorithmus mittels Wavelets entlang von Pfaden, bevor wir uns in Abschnitt 2.4 einem wesentlichen Bestandteil des Algorithmus, der Konstruktion geeigneter Pfade durch den vorliegenden Datensatz, widmen. In Abschnitt 2.5 werden wir einige theoretische Eigenschaften des Algorithmus untersuchen, während Abschnitt 2.6 Details über die Implementierung des Entstörungsalgorithmus und Abschnitt 2.7 numerische Resultate präsentieren werden. Abschließend werden in Abschnitt 2.8 mögliche Modifikationen des eingeführten Entstörungsalgorithmus diskutiert.

2.2 Wavelet-Filterbänke und Wavelet-Shrinkage

Als wirksames Werkzeug zur Zeit-Frequenz-Analyse wird die Wavelet-Transformation zur Entstörung und Kompression in der Signal- und Bildverarbeitung gerne genutzt.

Dabei zeichnet sich die Wavelet-Transformation gegenüber der verwandten Fourier-Transformation durch eine verbesserte Lokalisierung im Zeitbereich aus. Weiterhin profitiert die (diskrete) Wavelet-Transformation von der Existenz effizienter Algorithmen zur „schnellen Wavelet-Transformation“. Dieser Abschnitt soll in aller Kürze das allgemeine Prinzip einer Wavelet-Filterbank einführen. Wir beschränken uns auf eine Betrachtung der diskreten Wavelet-Transformation. Für weitere Ausführungen zur Wavelet-Transformation in der Signal- und Bildverarbeitung sei etwa auf die Bücher von Mallat [93] und Daubechies [37] verwiesen.

Einführende Definitionen

Wir halten zunächst den Begriff der Riesz-Basis sowie eine Notation für die Dilatation bzw. Translation einer Funktion fest, um anschließend Wavelets zu definieren.

Definition 2.48 • *Sei $\mathcal{H}$ ein Hilbertraum mit dem Skalarprodukt $\langle \cdot, \cdot \rangle$. Ein System $\{\psi_k\}_{k \in I} \subset \mathcal{H}$ mit einer höchstens abzählbaren Indexmenge I heißt Riesz-Basis von $\mathcal{H}$, wenn die folgenden Bedingungen gelten.*

1. *Die lineare Hülle von $\{\psi_k\}_{k \in I}$ liegt dicht in $\mathcal{H}$.*

2. *Es existieren Konstanten $0 < A \leq B < \infty$, sodass für jede Folge $c = (c_k)_{k \in I} \in l^2(I)$*

$$A\|c\|_{l^2(I)}^2 \leq \| \sum_{k \in I} c_k \psi_k \|_H^2 \leq B\|c\|_{l^2(I)}^2$$

erfüllt ist.

- *Wir nennen $\{\tilde{\psi}_k\}_{k \in I}$ die duale Riesz-Basis zur Riesz-Basis $\{\psi_k\}_{k \in I}$, falls $\{\tilde{\psi}_k\}_{k \in I}$ eine Riesz-Basis von $\mathcal{H}$ mit*

$$\langle \tilde{\psi}_k, \psi_{k'} \rangle = \delta_{k,k'} := \begin{cases} 1 & \text{für } k = k' \\ 0 & \text{sonst} \end{cases}$$

ist.

Bemerkung 2.49 *1. Jede Riesz-Basis besitzt eine eindeutig bestimmte duale Riesz-Basis.*

2. Riesz-Basen weichen das Konzept der Orthonormalbasen auf. Nach Eigenschaft 1 ist eine Riesz-Basis ein vollständiges Erzeugendensystem von $\mathcal{H}$. Eigenschaft 2 stellt eine Verallgemeinerung der Parsevalschen Ungleichung für Orthonormalbasen dar. Die Biorthogonalität der Elemente einer Riesz-Basis zu den Elementen

der zugehörigen dualen Riesz-Basis verallgemeinert die Orthonormalität der Elemente einer Orthonormalbasis.

Notation 2.50 *Für $\psi \in L^2(\mathbb{R})$ und $j, k \in \mathbb{Z}$ sei im Folgenden*

$$\psi_{j,k} := 2^{j/2}\psi(2^j \cdot -k).$$

Definition 2.51 *Eine Funktion $\psi \in L^2(\mathbb{R})$ heißt Wavelet (im Sinn der diskreten Wavelet-Transformation) genau dann, wenn*

- *$\{\psi_{j,k} : j, k \in \mathbb{Z}\}$ eine Riesz-Basis von $L^2(\mathbb{R})$ bildet, und*
- *eine Funktion $\tilde{\psi} \in L^2(\mathbb{R})$ existiert, sodass $\{\tilde{\psi}_{j,k} : j, k \in \mathbb{Z}\}$ die zugehörige duale Riesz-Basis von $L^2(\mathbb{R})$ darstellt.*

Bei den Funktionen ψ und $\tilde{\psi}$ spricht man auch von biorthogonalen Wavelets. Fallen ψ und $\tilde{\psi}$ zusammen, nennt man $\psi = \tilde{\psi}$ ein orthogonales Wavelet.

Bemerkung 2.52 *Mit gegebenen biorthogonalen Wavelets ψ und $\tilde{\psi}$ lässt sich eine Funktion $f \in L^2(\mathbb{R})$ in*

$$f = \sum_{j,k\in\mathbb{Z}} \langle f, \psi_{j,k}\rangle \tilde{\psi}_{j,k} = \sum_{j,k\in\mathbb{Z}} \langle f, \tilde{\psi}_{j,k}\rangle \psi_{j,k}$$

entwickeln.

Filterbänke perfekter Rekonstruktion

Wir gehen im Folgenden auf den Zusammenhang zwischen Wavelets und Filterbänken ein, welcher die Grundlage für die schnelle Wavelet-Transformation darstellt. Dazu erläutern wir zunächst, was wir unter einer Filterbank verstehen.

Abbildung 2.1 zeigt eine Zweikanal-Filterbank. Gegeben sei ein digitales Signal $c \in l^\infty(\mathbb{Z})$ sowie die Filter $h, g, \tilde{h}, \tilde{g} \in l^2(\mathbb{Z})$. Dabei bezeichnen $l^\infty(\mathbb{Z})$ und $l^2(\mathbb{Z})$ die üblichen Folgenräume, d.h.

$$l^\infty(\mathbb{Z}) := \{c = (c_k)_{k\in\mathbb{Z}} : \sup_{k\in\mathbb{Z}} |c_k| < \infty\}$$
$$l^2(\mathbb{Z}) := \{h = (h_k)_{k\in\mathbb{Z}} : \sum_{k\in\mathbb{Z}} |h_k|^2 < \infty\}.$$

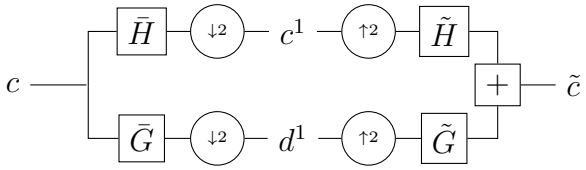

Abbildung 2.1: Eine Zweikanal-Filterbank.

Wir definieren die zugehörigen Symbole $H, G, \tilde{H}, \tilde{G}$ der Filter durch

$$\begin{aligned} H(\omega) &:= \sum_{k\in\mathbb{Z}} h_k e^{-i\omega k} \\ G(\omega) &:= \sum_{k\in\mathbb{Z}} g_k e^{-i\omega k} \\ \tilde{H}(\omega) &:= \sum_{k\in\mathbb{Z}} \tilde{h}_k e^{-i\omega k} \\ \tilde{G}(\omega) &:= \sum_{k\in\mathbb{Z}} \tilde{g}_k e^{-i\omega k}. \end{aligned}$$

Es ist ebenso üblich, die Symbole $H, G, \tilde{H}, \tilde{G}$ als Filter zu bezeichnen und die Folgen $h, g, \tilde{h}, \tilde{g}$ als Filterfolgen. Wir beschränken uns auf den Fall, in dem $h, g, \tilde{h}, \tilde{g}$ sogenannte FIR-Filter gemäß folgender Definition sind.

Definition 2.53 *Eine Folge $h = (h_k)_{k\in\mathbb{Z}} \in l^2(\mathbb{Z})$ heißt Filter mit endlicher Impulsantwort, kurz FIR-Filter (engl. finite impulse response filter), wenn nur endlich viele der Komponenten h_k von Null verschieden sind.*

Bemerkung 2.54 *Bei Termen wie $\sum_{k\in\mathbb{Z}} h_k$ handelt es sich im Folgenden daher eigentlich um endliche Summen, sodass sich die Frage nach Konvergenz dieser formal unendlichen Reihen nicht stellt.*

Zerlegungsseite einer Filterbank (Analyse) In der Zweikanal-Filterbank wird das Eingangssignal c zunächst mit dem Filter $(\bar{h}_{-k})_{k\in\mathbb{Z}}$ bzw. $(\bar{g}_{-k})_{k\in\mathbb{Z}}$ in Form einer Folgenfaltung gefiltert. Die Verwendung der komplex konjugierten Filter erfolgt dabei aus formalen Gründen. Man beachte, dass die Folgenfaltung einer Multiplikation im Symbolbereich entspricht. Auf die Resultate wird weiterhin der Downsampling-Operator $\downarrow 2 : l^\infty(\mathbb{Z}) \to l^\infty(\mathbb{Z})$ mit

$$\downarrow 2((c_k)_{k\in\mathbb{Z}}) := (c_{2k})_{k\in\mathbb{Z}}$$

für $(c_k)_{k\in\mathbb{Z}} \in l^\infty(\mathbb{Z})$ angewendet. Somit erhält man Folgen $c^1, d^1 \in l^\infty(\mathbb{Z})$ mit

$$c^1 = (\sum_{k\in\mathbb{Z}} c_k \bar{h}_{k-2n})_{n\in\mathbb{Z}}$$
$$d^1 = (\sum_{k\in\mathbb{Z}} c_k \bar{g}_{k-2n})_{n\in\mathbb{Z}}.$$

In der Anwendung werden die Filter so gewählt, dass die Filterung eine Trennung des Signals in Hoch- und Tiefpassanteil bewirkt. Das Downsampling bewirkt ein Löschen jeder zweiten Komponente der Hoch- bzw. Tiefpasskoeffizienten.

Rekonstruktionsseite einer Filterbank (Synthese) Soweit wurde die Zerlegungsseite einer Filterbank beschrieben. Es folgt die Rekonstruktionsseite. Zunächst wendet man den Upsampling-Operator $\uparrow 2 : l^\infty(\mathbb{Z}) \to l^\infty(\mathbb{Z})$ mit

$$\uparrow 2((c_k)_{k\in\mathbb{Z}}) = \begin{cases} c_k & \text{falls } k \text{ gerade} \\ 0 & \text{falls } k \text{ ungerade} \end{cases}$$

für $(c_k)_{k\in\mathbb{Z}} \in l^\infty(\mathbb{Z})$ auf die Tiefpasskoeffizienten c^1 bzw. die Hochpasskoeffizienten d^1 an. Danach erfolgt eine Filterung mit dem Filter $\tilde{h}$ bzw. $\tilde{g}$. Die resultierenden Folgen werden addiert, um eine Folge $\tilde{c} \in l^\infty(\mathbb{Z})$ zu erhalten, d.h.

$$\tilde{c} = (\sum_{k\in\mathbb{Z}} c_k^1 \tilde{h}_{n-2k} + \sum_{k\in\mathbb{Z}} d_k^1 \tilde{g}_{n-2k})_{n\in\mathbb{Z}}.$$

Dieses Signal $\tilde{c}$ stellt eine - möglicherweise fehlerbehaftete - Rekonstruktion des Ausgangssignals c dar. Wir interessieren uns für sogenannte Filterbänke perfekter Rekonstruktion, welche eine fehlerlose Rekonstruktion, die sich bei speziellen Konstellationen der Filter ergibt, liefern.

Definition 2.55 *Eine Filterbank, wie bisher beschrieben, heißt Filterbank perfekter Rekonstruktion, falls*

$$c = \tilde{c}$$

für jedes Eingangssignal $c \in l^\infty(\mathbb{Z})$ gilt.

Iterative Filterbänke und Zusammenhang mit Wavelets Die bisherige Betrachtung umfasst nur ein einziges Zerlegungslevel. Üblicherweise verwendete man mehrere Zerlegungslevel, welche man durch iterative Anwendung der Filterbank auf die Folgen der

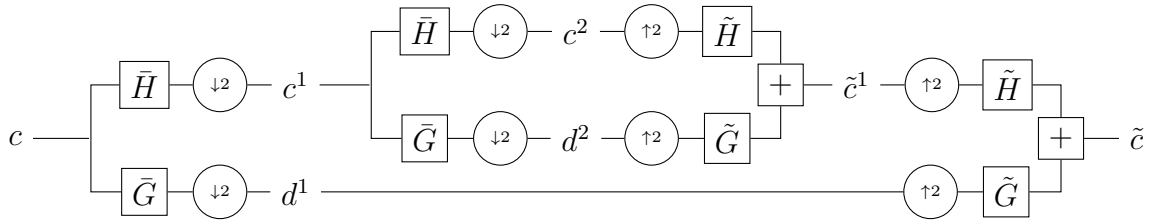

Abbildung 2.2: Eine iterative Filterbank mit zwei Zerlegungsleveln.

Tiefpass-Koeffizienten erreicht. Abbildung 2.2 illustriert eine iterative Filterbank mit zwei Zerlegungsleveln.

Es besteht ein direkter Zusammenhang zwischen Filterbänken perfekter Rekonstruktion und Wavelets gemäß des folgenden von Cohen et al. bewiesenen Satzes (vgl. [30]).

Satz 2.56 *Gegeben sei eine aus FIR-Filtern $H, G, \tilde{H}, \tilde{G}$ mit der Normierung*

$$\sum_{k\in\mathbb{Z}} h_k = \sum_{k\in\mathbb{Z}} \tilde{h}_k = \sqrt{2}$$

bestehende Filterbank perfekter Rekonstruktion. Zusätzlich seien die beiden folgenden Bedingungen 1 und 2 erfüllt.

1. *Es gebe strikt positive trigonometrische Polynome P und $\tilde{P}$, sodass*

$$|H\left(\frac{\omega}{2}\right)|^2 P\left(\frac{\omega}{2}\right) + |H\left(\frac{\omega}{2}+\pi\right)|^2 P(\frac{\omega}{2}+\pi) = 2P(\omega)$$
$$|\tilde{H}\left(\frac{\omega}{2}\right)|^2 \tilde{P}\left(\frac{\omega}{2}\right) + |\tilde{H}\left(\frac{\omega}{2}+\pi\right)|^2 \tilde{P}(\frac{\omega}{2}+\pi) = 2\tilde{P}(\omega)$$

   für alle $\omega \in [-\pi, \pi)$ gilt.

2. *Es gelte*

$$\inf_{[-\frac{\pi}{2},\frac{\pi}{2}]} |H(\omega)| > 0$$
$$\inf_{[-\frac{\pi}{2},\frac{\pi}{2}]} |\tilde{H}(\omega)| > 0.$$

Es existieren dann sogenannte Skalierungsfunktionen $\phi, \tilde{\phi} \in L^2(\mathbb{R})$, deren Fourier-Transformierte den Funktionalgleichungen

$$\hat{\phi}(\omega) = \frac{1}{\sqrt{2}} H\left(\frac{\omega}{2}\right) \hat{\phi}\left(\frac{\omega}{2}\right)$$
$$\hat{\tilde{\phi}}(\omega) = \frac{1}{\sqrt{2}} \tilde{H}\left(\frac{\omega}{2}\right) \hat{\tilde{\phi}}\left(\frac{\omega}{2}\right)$$

genügen. Weiterhin sind die gemäß

$$\hat{\psi}(\omega) = \frac{1}{\sqrt{2}} G\left(\frac{\omega}{2}\right) \hat{\phi}\left(\frac{\omega}{2}\right)$$
$$\hat{\tilde{\psi}}(\omega) = \frac{1}{\sqrt{2}} \tilde{G}\left(\frac{\omega}{2}\right) \hat{\tilde{\phi}}\left(\frac{\omega}{2}\right)$$

definierten Funktionen $\psi, \tilde{\psi} \in L^2(\mathbb{R})$ biorthogonale Wavelets.

Beispiel 2.57 *Wir betrachten drei Beispiele für Wavelet-Filterbänke. Die angegebenen Filterkoeffizienten weisen dabei jeweils die Normierung $\sum_{k\in\mathbb{Z}} h_k = \sqrt{2}$ auf.*

1. *Das einfachste und älteste Beispiel für ein Wavelet, stellt das von Haar [60] eingeführte Haar-Wavelet dar. Das Haar-Wavelet ist ein orthogonales Wavelet. Filterung eines Signals mit dem Haar-Tiefpass-Filter liefert (bis auf Vorfaktor) arithmetische Mittelwerte zweier aufeinanderfolgender Signalwerte. Filterung mit dem Haar-Hochpass-Filter liefert (bis auf Vorfaktor) Differenzen zweier aufeinanderfolgender Signalwerte. Die Skalierungsfunktion ϕ des Haar-Wavelets ist die Indikatorfunktion des Einheitsintervalls $[0, 1)$, d.h.*

$$\phi(x) = \begin{cases} 1 & \text{falls } 0 \le x < 1 \\ 0 & \text{sonst} \end{cases},$$

und die zugehörige Wavelet-Funktion ψ ist die stückweise konstante Funktion

$$\psi(x) = \begin{cases} 1 & \text{falls } 0 \le x < \frac{1}{2} \\ -1 & \text{falls } \frac{1}{2} \le x < 1 \\ 0 & \text{sonst} \end{cases}.$$

Die zugehörigen Filter $h = \tilde{h}$ und $g = \tilde{g}$ sind in Tabelle 2.1 gegeben.

2. *Sehr bewährt für die Anwendung in der Bildentstörung haben sich die symmetrischen, biorthogonalen Cohen-Daubechies-Feauveau-9/7-Filter (CDF-9/7-Filter) [30] mit Filterlängen 9 bzw. 7. Numerische Werte für die Filterkoeffizienten sind in Tabelle 2.2 aufgelistet.*

3. *Ein weiteres Beispiel für in der Bildverarbeitung verbreitete Filter sind die nicht symmetrischen, orthogonalen Daubechies-Wavelets [37], z.B. die D4-Filter. Hier verstehen wir unter „D4“ die Daubechies-Filter der Länge 4. In Bezugnahme darauf, dass der Hochpass-Filter zwei verschwindende Momente besitzt, werden diese Filter zuweilen auch unter dem Namen „D2“ geführt. Tabelle 2.3 führt die Koeffizienten der Filter $h = \tilde{h}$ und $g = \tilde{g}$ auf.*

k	$\sqrt{2}h_k$	$\sqrt{2}g_k$
<0	0	0
0	1	1
1	1	-1
>1	0	0

Tabelle 2.1: Koeffizienten der Haar-Filter.

k	$h_k/\sqrt{2}$	$g_k/\sqrt{2}$	$\tilde{h}_k/\sqrt{2}$	$\tilde{g}_k/\sqrt{2}$
<-4	0	0	0	0
-4	0	0.026748757411	0.026748757411	0
-3	-0.045635881557	-0.016864118443	0.016864118443	0.045635881557
-2	-0.028771763114	-0.078223266529	-0.078223266529	-0.028771763114
-1	0.295635881557	0.266864118443	-0.266864118443	-0.295635881557
0	0.557543526229	0.602949018236	0.602949018236	0.557543526229
1	0.295635881557	0.266864118443	-0.266864118443	-0.295635881557
2	-0.028771763114	-0.078223266529	-0.078223266529	-0.028771763114
3	-0.045635881557	-0.016864118443	0.016864118443	0.045635881557
4	0	0.026748757411	0.026748757411	0
>4	0	0	0	0

Tabelle 2.2: Numerische Werte für die Koeffizienten der CDF-9/7-Filter (siehe [30]).

k	$4\sqrt{2}h_k$	$4\sqrt{2}g_k$
<-2	0	0
-2	0	$1-\sqrt{3}$
-1	0	$-3+\sqrt{3}$
0	$1+\sqrt{3}$	$3+\sqrt{3}$
1	$3+\sqrt{3}$	$-1-\sqrt{3}$
2	$3-\sqrt{3}$	0
3	$1-\sqrt{3}$	0
>2	0	0

Tabelle 2.3: Koeffizienten der D4-Filter (siehe etwa [119]).

Bemerkung 2.58 *1. In der Anwendung arbeiten wir mit endlichen Signalen (Vektoren) $c \in \mathbb{R}^N$, $N \in \mathbb{N}$, statt mit unendlichen Signalen (Folgen) $c \in l^\infty(\mathbb{Z})$. Es erfolgt in diesem Fall eine Zerlegung von c in einen Tiefpassanteil $c^1 \in \mathbb{R}^{N/2}$ und einen Hochpassanteil $d^1 \in \mathbb{R}^{N/2}$. Wir nutzen zur Vereinfachung trotzdem Schreibweisen wie*

$$c_n^1 = \sum_{k\in\mathbb{Z}} c_k \bar{h}_{k-2n}$$
$$d_n^1 = \sum_{k\in\mathbb{Z}} c_k \bar{g}_{k-2n}$$

für $n = 1, \ldots, N/2$ mit geeigneten Konventionen (Randbedingungen) für die Werte von c_k für $k \notin \{1, \ldots, N\}$.

2. Unter Verwendung sogenannter periodischer Filter lässt sich für ein Eingangssignal $c \in \mathbb{R}^N$ ein Filterbank-Schritt mittels einer Transformationsmatrix $W \in \mathbb{R}^{N\times N}$ auch in Matrixschreibweise

$$Wc = \begin{pmatrix} c^1 \\ d^1 \end{pmatrix}$$

schreiben.

Wavelet-Shrinkage

Die Grundidee, Entstörung eines Signals mittels einer Wavelet-Transformation zu erreichen, liegt in der Annahme begründet, dass Störungen hochfrequent sind und sich somit in betragsmäßig kleinen Hochpass-Koeffizienten äußern. Nach der Zerlegung des Signals werden daher betragsmäßig kleine Hochpass-Koeffizienten mittels eines Shrinkage-Operators zu Null gesetzt. Anschließend wird das Signal aus den (unveränderten) Tiefpasskoeffizienten und den (modifizierten) Hochpasskoeffizienten rekonstruiert (siehe auch Abbildung 2.3). Als Shrinkage-Operator nutzt man etwa den Hard-Shrinkage-Operator $S_\theta : l^\infty(\mathbb{Z}) \to l^\infty(\mathbb{Z})$ mit

$$(S_\theta(d))_k = \begin{cases} d_k & \text{falls } |d_k| \geq \theta \\ 0 & \text{falls } |d_k| < \theta \end{cases}$$

für $d \in l^\infty(\mathbb{Z})$ und einem fixierten Shrinkage-Parameter $\theta > 0$. Der Parameter θ muss dabei für das jeweils vorliegende Signal und Rauschen sorgfältig gewählt werden, sodass Störungen möglichst gut eliminiert werden, ohne zu viele Signaldetails zu verlieren.

Wavelet-Shrinkage arbeitet sehr gut bei der Beseitigung von additivem Gaußschen Rauschen. Für davon abweichende Arten von Rauschen sind andere Entstörungsansätze erforderlich.

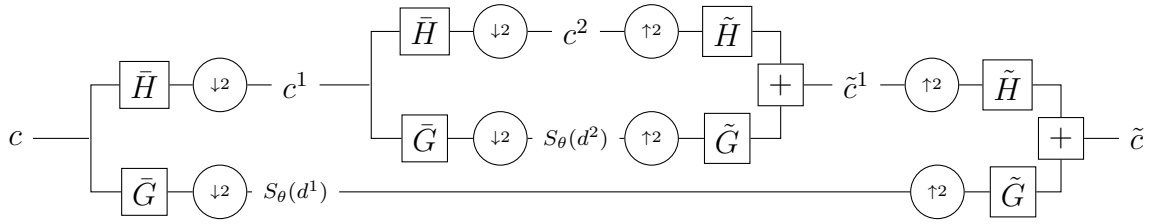

Abbildung 2.3: Eine Filterbank mit Shrinkage-Operator.

Zweidimensionale Wavelet-Transformation

Bisher haben wir nur eindimensionale Wavelets betrachtet. Die einfachste Variante, eine Wavelet-Transformation für Bilder (d.h. zweidimensionale Signale) einzuführen, benutzt Tensorprodukte eindimensionaler Wavelets und Skalierungsfunktionen. Seien dafür $\psi, \tilde{\psi} \in L^2(\mathbb{R})$ eindimensionale biorthogonale Wavelets mit zugehörigen Skalierungsfunktionen $\phi, \tilde{\phi} \in L^2(\mathbb{R})$. Mit den Tensorprodukten

$$\begin{aligned} \psi^1(x,y) &= \phi(x)\psi(y) & \tilde{\psi}^1(x,y) &= \tilde{\phi}(x)\tilde{\psi}(y) \\ \psi^2(x,y) &= \psi(x)\phi(y) & \tilde{\psi}^2(x,y) &= \tilde{\psi}(x)\tilde{\phi}(y) \\ \psi^3(x,y) &= \psi(x)\psi(y) & \tilde{\psi}^3(x,y) &= \tilde{\psi}(x)\tilde{\psi}(y) \end{aligned}$$

konstruieren wir die zweidimensionalen Tensorprodukt-Wavelets

$$\begin{aligned} \psi^\nu_{j,k_1,k_2}(x,y) &= 2^j \psi^\nu(2^j x - k_1, 2^j y - k_2) \\ \tilde{\psi}^\nu_{j,k_1,k_2}(x,y) &= 2^j \tilde{\psi}^\nu(2^j x - k_1, 2^j y - k_2) \end{aligned}$$

für $\nu = 1,2,3,\ j,k_1,k_2 \in \mathbb{Z}$. Es lässt sich zeigen, dass $\{\psi^\nu_{j,k_1,k_2}\}_{\nu\in\{1,2,3\},j,k_1,k_2\in\mathbb{Z}}$ und $\{\tilde{\psi}^\nu_{j,k_1,k_2}\}_{\nu\in\{1,2,3\},j,k_1,k_2\in\mathbb{Z}}$ biorthogonale Riesz-Basen von $L^2(\mathbb{R}^2)$ sind.

Bei der diskreten zweidimensionalen Wavelet-Transformation erfolgt eine Zerlegung des Eingangsbildes $c \in \mathbb{R}^{N_1\times N_2}$ in den Tiefpassanteil $c_1 \in \mathbb{R}^{N_1/2\times N_2/2}$ und drei Hochpassanteile $d_1^1, d_1^2, d_1^3 \in \mathbb{R}^{N_1/2\times N_2/2}$. Eine schnelle Wavelet-Transformation mittels des Konzepts der Filterbänke erhält man wie folgt: In einem ersten Schritt wendet man die entsprechenden Filter der zugehörigen eindimensionalen Wavelet-Filterbank auf die Spalten des Eingangsbildes an. Anschließend wendet man die entsprechenden eindimensionalen Filter auf die Zeilen des Resultats des ersten Schrittes an.

Cycle-Spinning

Eine Methode, translationsinvariante Entstörung oder Cycle-Spinning genannt, mit der man u.a. die Bildentstörung mittels Wavelets durch Ausnutzung von Redundanz wesentlich verbessern kann, findet sich in der Arbeit [31] von Coifman und Donoho. Die

Funktionsweise des Cycle-Spinnings im Rahmen der Entstörung eines gestörten Bildes $\tilde{Y} = Y + Z \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}}$ mit additivem Gaußschen Rauschen Z ist im folgenden Algorithmus dargelegt.

Algorithmus 2.59 Zweidimensionale Tensorprodukt-Wavelet-Transformation mit Cycle-Spinning
Seien $S_1, S_2 \in \{0, \ldots, \sqrt{N}\}$.

- *Für $s_1 = 0, \ldots, S_1$ und $s_2 = 0, \ldots, S_2$:*
  - *Verschiebe die Pixel des Bildes $\tilde{Y}$ zyklisch um den Vektor (s_1, s_2).*
  - *Führe eine Wavelet-Shrinkage-Prozedur am „verschobenen“ Bild durch.*
  - *Mache die Verschiebung rückgängig, um ein entstörtes Bild $\hat{Y}_{(s_1,s_2)}$ zu erhalten.*
- *Die finale Rekonstruktion $\hat{Y}$ des entstörten Bildes ergibt sich durch Durchschnittsbildung, d.h.*

$$\hat{Y} = \frac{1}{S_1+1}\frac{1}{S_2+1}\sum_{s_1=0}^{S_1}\sum_{s_2=0}^{S_2}\hat{Y}_{(s_1,s_2)}.$$

Bemerkung 2.60 *Üblicherweise setzt man $S_1 = S_2 = 7$, sodass am Ende der Durschnitt über 64 entstörte Bilder betrachtet wird.*

2.3 Ein Algorithmus zur Entstörung mittels Wavelets entlang von Pfaden

Wir gehen von einer Menge

$$\Gamma = \{x_1, \ldots, x_N\}$$

gestreuter Punkte $x_j \in \mathbb{R}^d$ mit $d \in \mathbb{N}_{\geq 2}$ aus. Dabei sei Γ in einer Menge $\Omega \subset \mathbb{R}^d$ enthalten, welche sich als Vereinigung endlich vieler beschränkter, zusammenhängender Teilmengen des $\mathbb{R}^d$ schreiben lässt. Weiterhin sind uns für $j = 1, \ldots, N$ die gestörten Abtastwerte

$$\tilde{f}(x_j) = f(x_j) + z_j$$

gegeben. Dabei seien $f : \Omega \to \mathbb{R}$ eine stückweise stetige Funktion und z_j unabhängige normalverteilte Zufallsvariablen mit Erwartungswert 0 und unbekannter Varianz σ^2. Die

abgetasteten Funktionswerte sind demnach mit additivem Gaußschen Rauschen belegt. Ziel ist eine Entstörung der Daten, d.h. eine möglichst gute Rekonstruktion der originalen Abtastwerte $f(x_j)$ für $j = 1, \ldots, N$.

Wir nehmen an, dass Γ quasi-uniform gemäß der folgenden Definition (siehe [10]) ist.

Definition 2.61 *Erfüllen die maximale Dichte (engl. maximal density)*

$$\delta(\Gamma) := \max_{x \in \Omega} \min_{x_k \in \Gamma} \|x - x_k\|_2$$

und der minimale Abstand (engl. minimal spacing)

$$\mu(\Gamma) := \min_{\substack{x_j, x_k \in \Gamma, \\ j \neq k}} \|x_j - x_k\|_2$$

die Beziehung

$$\delta(\Gamma) < C \cdot \mu(\Gamma)$$

mit einer a priori fixierten Konstanten C passender Größe (siehe auch Bemerkung 2.62), so heißt Γ quasi-uniform in Ω.

Bemerkung 2.62

1. *Die Quasi-Uniformität stellt sicher, dass die Menge Ω überall „genügend dicht“ mit Abtastpunkten $x_j \in \Gamma$ abgedeckt ist. Die maximale Dichte $\delta(\Gamma)$ gibt den maximalen Fehler an, den man bei der Bestapproximation eines Punktes $x \in \Omega$ durch einen der Abtastpunkte $x_j \in \Gamma$ begeht.*

2. *Betrachten wir im Fall $\sqrt[d]{N} \in \mathbb{N}$ etwa den d-dimensionalen Würfel $\Omega = [1, \sqrt[d]{N}]^d$ und versehen ihn mit einem regulären Gitter, d.h. $\Gamma = \{(\iota_1, \ldots, \iota_d)^T : \iota_1, \ldots, \iota_d \in \{1, \ldots, \sqrt[d]{N}\}\}$, so gilt $\mu(\Gamma) = 1$, $\delta(\Gamma) = \sqrt{d}/2$ und folglich*

$$\delta(\Gamma) = \frac{\sqrt{d}}{2} \mu(\Gamma).$$

3. *Die Autoren in [10] verwenden $C = 2$ unabhängig von der Raumdimension d in der Definition der Quasi-Uniformität. In diesem Fall ist ein uniformes Gitter für $d \geq 16$ per Definition nicht mehr quasi-uniform (siehe vorherigen Punkt dieser Bemerkung). Hier sei daher $C > \sqrt{d}/2$. Es kann sinnvoll sein, auch für $d < 16$ Werte für C zuzulassen, die etwas größer als 2 sind.*

Der im Folgenden diskutierte und von uns in [66] neu vorgeschlagene Entstörungsalgorithmus bedient sich einer klassischen Wavelet-Shrinkage-Prozedur, wobei die Wavelet-Transformation entlang zu konstruierender Pfade erfolgt, und stellt somit eine Verallgemeinerung der EPWT [101] dar. In Anlehnung an das Cycle-Spinning [31] verwenden

wir eine Durchschnittsbildung über mehrere mittels unterschiedlicher Pfade entstörte Datensätze, um das Resultat der Entstörung zu verbessern.

Zusammenfassen lässt sich unser neuer Entstörungsalgorithmus wie folgt (vgl. [66]).

Algorithmus 2.63 Entstörung mittels Wavelets entlang von Pfaden
Gegeben: $\Gamma = \{x_1, \ldots, x_N\} =: \{x_1^0, \ldots, x_N^0\} =: \Gamma^0 \subset \mathbb{R}^d$, $\tilde{f}(x_j) =: c_j^0$ *für* $j = 1, \ldots, N$, *wobei* $N = 2^t$ *für ein* $t \in \mathbb{N}$ *und eine biorthogonale Wavelet-Filterbank mit Zerlegungsfiltern* h, g *und Rekonstruktionsfiltern* $\tilde{h}, \tilde{g}$ *mit* $\sum_{k\in\mathbb{Z}} h_k = \sqrt{2}$ *sowie ein Tiefpass-Filter* γ *mit* $\sum_{k\in\mathbb{Z}} \gamma_k = 1$.

Zerlegung: *Führe die folgenden vier Schritte für* $l = 0, 1, \ldots, L-1$ *mit* $L < t$ *durch:*

1. *Konstruiere einen geeigneten Pfadvektor* $p^l \in \mathbb{R}^{N/2^l}$, *d.h. eine Permutation von* $\{1, \ldots, N/2^l\}$, *welche eine Reihenfolge der Punkte* $x^l_{p^l(j)}$ *und der zugeordneten Werte* $c^l_{p^l(j)}$ *darstellt.*

2. *Wende den (periodischen) Tiefpass-Filter* $(\bar{h}_{-k})_{k\in\mathbb{Z}}$ *bzw. den (periodischen) Hochpass-Filter* $(\bar{g}_{-k})_{k\in\mathbb{Z}}$ *auf den Vektor* $(c^l_{p^l(j)})_{j=1}^{N/2^l}$ *an und erhalte nach Downsampling die Tiefpass-Koeffizienten* $(c_j^{l+1})_{j=1}^{N/2^{l+1}}$ *bzw. die Hochpass-Koeffizienten* $(d_j^{l+1})_{j=1}^{N/2^{l+1}}$ *mit*
$$c_j^{l+1} = \sum_{k\in\mathbb{Z}} c^l_{p^l(k)} \bar{h}_{k-2j+1}$$
$$d_j^{l+1} = \sum_{k\in\mathbb{Z}} c^l_{p^l(k)} \bar{g}_{k-2j+1}.$$

3. *Wende den Tiefpass-Filter* $(\bar{\gamma}_{-k})_{k\in\mathbb{Z}}$ *auf den Vektor* $(x^l_{p^l(j)})_{j=1}^{N/2^l}$ *(separat für jede der d Komponenten) an und erhalte nach Downsampling neue gestreute Punkte* $(x_j^{l+1})_{j=1}^{N/2^{l+1}}$ *mit*
$$x_j^{l+1} = \sum_{k\in\mathbb{Z}} \bar{\gamma}_{k-2j+1} x_{p^l(k)}.$$
Setze $\Gamma^{l+1} := \{x_1^{l+1}, \ldots, x^{l+1}_{N/2^{l+1}}\}$.

4. *Führe ein Hard-Shrinkage mit einem Parameter* $\theta > 0$ *an den Hochpass-Koeffizienten* $(d_j^{l+1})_{j=1}^{N/2^{l+1}}$ *resultierend in*
$$\tilde{d}_j^{l+1} = \begin{cases} d_j^{l+1} & \text{falls } |d_j^{l+1}| \geq \theta \\ 0 & \text{falls } |d_j^{l+1}| < \theta \end{cases}$$
durch.

Rekonstruktion: *Initialisiere* $(\tilde{c}_j^L)_{j=1}^{N/2^L} := (c_j^L)_{j=1}^{N/2^L}$ *und führe die folgenden drei Schritte für* $l = L, L-1, \ldots, 1$ *durch:*

5. *Wende Upsampling und anschließend den Tiefpass-Filter* $\tilde{h}$ *auf* $(\tilde{c}_j^l)_{j=1}^{N/2^l}$ *an.*

6. *Wende Upsampling und anschließend den Hochpass-Filter* $\tilde{g}$ *auf* $(\tilde{d}_j^l)_{j=1}^{N/2^l}$ *an.*

7. *Addiere die Ergebnisse von Schritt 5 und 6, um*

$$(\tilde{c}^{l-1}_{p^{l-1}(j)})_{j=1}^{N/2^{l-1}} = \left(\sum_{k\in\mathbb{Z}} \tilde{c}_k^l \tilde{h}_{j-2k+1} + \sum_{k\in\mathbb{Z}} \tilde{d}_k^l \tilde{g}_{j-2k+1} \right)_{j=1}^{N/2^{l-1}}$$

zu erhalten. Invertiere die Permutation mit dem Resultat $(\tilde{c}_j^{l-1})_{j=1}^{N/2^{l-1}}$.

Ausgabe: *Den gestreuten Punkten* $x_j \in \Gamma$ *zugeordnete geglättete Funktionswerte* $(\tilde{c}_j^0)_{j=1}^N$.

Iteration: *Wiederhole die Schritte 1 bis 7 (unter Verwendung neuer Pfadvektoren)* T*-mal (z.B.* $T = 64$*) und bilde den Durchschnitt der Ausgaben* $(\tilde{c}_j^0)_{j=1}^N$.

Bemerkung 2.64 *1. Die Konstruktion geeigneter Pfade ist wesentlich für den Erfolg des Algorithmus. Wir werden im nächsten Abschnitt 2.4 erläutern, wie wir die Pfade festlegen. Man beachte, dass wir in jedem Zerlegungslevel einen neuen Pfad bestimmen. Dadurch ist die im obigen Entstörungsalgorithmus verwendete Transformation nicht echt eindimensional.*

2. *Schritt 3 des Algorithmus aktualisiert in jedem Zerlegungslevel die zugrundeliegende Punktmenge* Γ^l *durch* Γ^{l+1}*. Durch das Downsampling findet eine Ausdünnung in Form einer Halbierung der Kardinalität der Punkmenge statt, d.h.*

$$|\Gamma^{l+1}| = \frac{1}{2}|\Gamma^l| = \frac{N}{2^{l+1}}.$$

Die Wahl $\gamma = \frac{1}{\sqrt{2}}h$ *ist naheliegend, aber andere Tiefpass-Filter sind möglich. Man beachte, dass, wenn* $\Gamma = \Gamma^0 \subset \Omega \subset \mathbb{R}^d$ *ist, für* $l \geq 1$ *trotz der Normierung* $\sum_{k\in\mathbb{Z}} \gamma_k = 1$ *nicht notwendigerweise* $\Gamma^l \subset \Omega$ *folgt. Für die numerischen Resultate in Abschnitt 2.7 hat es sich als ausreichend erwiesen, für die Filterung der Punktmenge* $\Gamma^l := \{x_1^l, \ldots, x_{N/2^l}^l\}$ *ein reines Downsampling entlang des Pfades zu verwenden. Den zugehörigen trivialen „Filter“ bezeichnet man zuweilen auch als „Lazy-Filter“.*

3. *Man beachte, dass wir im Gegensatz zur ursprünglichen EPWT in jedem Zerlegungslevel mit gestreuten Punkten* $x_j^l \in \mathbb{R}^d$ *arbeiten. Die „Punkte“, entlang derer*

Pfade konstruiert werden, sind bei der EPWT in [101] ab dem zweiten Zerlegungslevel sogenannte Indexmengen, welche sich als Vereinigung zweier im Pfad des vorherigen Level aufeinanderfolgender Punkte bzw. Indexmengen ergeben. In [130] wird von Tenorth auch eine sogenannte „Mittelpunkt-EPWT“ diskutiert, bei der eine Indexmenge durch ihren Mittelpunkt identifiziert wird. Dieses Vorgehen entspricht Algorithmus 2.63 unter Verwendung eines entsprechend normierten Haar-Filters für die Aktualisierung der Punktmenge Γ^l.

4. *Für das Thresholding in Schritt 4 sind andere Shrinkage-Funktionen wie Soft-, Firm- oder Garrote-Shrinkage (siehe etwa [96]) denkbar. Die Wahl des Shrinkage-Parameters* θ *hat entscheidenden Einfluss auf das Resultat der Entstörung und muss daher sorgfältig erfolgen.*

5. *Die einzelnen Iterationen des Algorithmus sind vollkommen unabhängig voneinander. Der Gesamtaufwand des Algorithmus steigt folglich linear mit der Anzahl der verwendeten Iterationen. Andererseits ermöglicht die Unabhängigkeit der Iterationen eine Parallelisierung. Damit kann der zeitliche Aufwand bei der Durchführung des Algorithmus erheblich reduziert werden.*

6. *Der Algorithmus 2.63 ist grundsätzlich für beliebige Dimension* $d \in \mathbb{N}_{\geq 2}$ *anwendbar. Außer der Voraussetzung, dass die Punktmenge* Γ *quasi-uniform (siehe Definition 2.61) in einer Menge* Ω*, welche sich als Vereinigung endlich vieler beschränkter, zusammenhängender Teilmengen des* $\mathbb{R}^d$ *schreiben lässt, ist, bestehen weiterhin für die Anwendbarkeit von Algorithmus 2.63 keinerlei Einschränkungen an die Form und Verteilung der Punktmenge. Insbesondere sind wir nicht auf zweidimensionale, reguläre und rechteckige Gitter angewiesen. Der Pfad (im Sinne eines Polygonzuges von* $x^l_{p^l(1)}$ *nach* $x^l_{p^l(N/2^l)}$*) muss, wenn* Ω *etwa nicht zusammenhängend oder nicht konvex ist, im Allgemeinen nicht vollständig in* Ω *enthalten sein, was für die Durchführung des Algorithmus 2.63 grundsätzlich jedoch nicht von Bedeutung ist.*

7. *Algorithmus 2.63 lässt sich wie in Abbildung 2.4 als modifizierte Wavelet-Filterbank interpretieren. Dabei wird vor der Anwendung der Zerlegungsfilter ein Permutationsoperator* P_l *eingefügt, welcher die Tiefpass-Daten* $(c^l_j)_{j=1}^{N/2^l}$ *zu einer Folge* $(c^l_{p^l(j)})_{j=1}^{N/2^l}$ *entlang des konstruierten Pfades* p^l *umordnet. Analog wird nach der Rekonstruktion der Tiefpass-Werte der zugehörige inverse Permuationsoperator* $(P_l)^{-1}$ *ergänzt. Man beachte, dass bei der Darstellung in Abbildung 2.4 jedoch die Adaptivität der Permutationsoperatoren nicht deutlich wird. Weiterhin sind die im Hintergrund ablaufenden Aktualisierungen der Punktmengen* Γ^l *nicht dargestellt.*

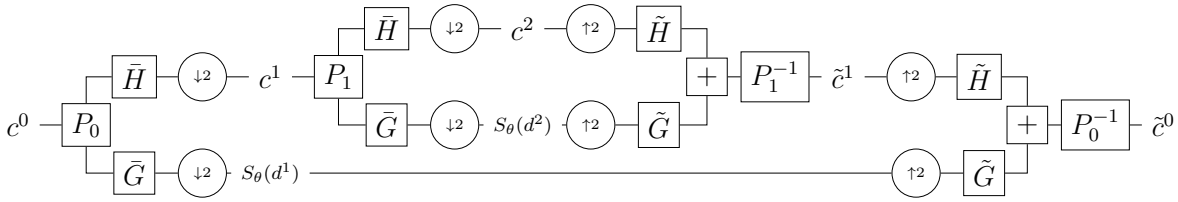

Abbildung 2.4: Algorithmus 2.63 als modifizierte Filterbank (vgl. Abbildungen in [108]).

2.4 Adaptive Pfadkonstruktionen

Entscheidend für den Erfolg des Entstörungsalgorithmus 2.63 ist die Konstruktion passender Pfade auf dem Datensatz. Die Pfade sollen gewissen Strukturen innerhalb des Datensatzes folgen und sind daher adaptiv zu wählen. Die für die Bildapproximation eingeführte EPWT [101] nutzt die Korrelation von Bildwerten benachbarter Bildpunkte. Die Pfade für die EPWT werden rekursiv konstruiert. Ein neuer Punkt wird bei der ursprünglichen, sogenannten rigorosen (engl. rigorous), EPWT als derjenige Nachbarpunkt des aktuellen Punktes gewählt, dessen Bildwert die geringste Differenz zum Bildwert des aktuellen Punktes aufweist. Im Falle verrauschter Daten sind die Korrelationen der Funktionswerte jedoch beeinträchtigt. Folglich müssen wir die Pfadkonstruktion modifizieren.

Man beachte, dass die zu konstruierenden Pfade jeden Punkt x_j^l in Γ^l genau einmal besuchen. Wir können die Pfade daher mit Pfadvektoren $(p^l(1), \ldots, p^l(N/2^l))$, d.h. Permutationen der Indizes $1, \ldots, N/2^l$ der Punkte $x_1^l, \ldots, x_{N/2^l}^l$, identifizieren. Wir bezeichnen daher im Folgenden äquivalent die Punkte als auch die zugehörigen Indizes als Bestandteile des Pfades.

In den folgenden zwei Unterabschnitten schlagen wir jeweils eine Pfadkonstruktion vor, welche wir adaptiv deterministisch (engl. adaptive deterministic) bzw. adaptiv zufällig (engl. adaptive random) (siehe [66]) nennen werden.

2.4.1 Adaptiv deterministische Pfadkonstruktion

Wir konstruieren von einem beliebigen Startpunkt aus rekursiv einen Pfad durch die Menge der Punkte $\Gamma^l = \{x_1^l, \ldots, x_{N/2^l}^l\} \subset \mathbb{R}^d$ des l-ten Level, wobei $0 \leq l \leq L-1$ sei (siehe Algorithmus 2.63). Dabei sollen sich aufeinanderfolgende Punkte des Pfades *ähnlich* sein. Wir messen die Ähnlichkeit einerseits mittels des spatialen Abstandes der Punkte und andererseits mittels der Differenz der zugeordneten Funktionswerte (bzw.

der zugeordneten Tiefpass-Werte in höheren Leveln). Dazu definieren wir zunächst die (spatiale) Nachbarschaft eines Punkte $x_j^l \in \Gamma^l \subset \mathbb{R}^d$ gemäß

$$N_{C_1}(x_j^l) := \{x_k^l \in \Gamma^l : \|x_j^l - x_k^l\|_2 \leq 2^{l/d} C_1, j \neq k\} \tag{2.1}$$

mit einer Konstanten $C_1 > 0$. Der Faktor $2^{l/d}$ wurde eingefügt, da sich die Anzahl der Punkte von Level zu Level halbiert und sich somit die Distanzen zwischen den Punkten vergrößern.

Bemerkung 2.65 *Ein alternatives Konzept, welches für eine fixierte natürliche Zahl $n < N/2^l$ die Nachbarschaft eines Referenzpunktes als die Menge der n Punkte mit dem geringsten spatialen Abstand zum Referenzpunkt festlegt, ist denkbar (vgl. auch Kapitel 1, Unterabschnitt 1.1.3). Insbesondere bei der Anwendung auf Bilder mit auf einem regulären Rechteckgitter verteilten Pixeln kann die Nutzung einer klassischen Achter-Nachbarschaft (Moore-Nachbarschaft), bei der die Nachbarn eines Pixels als die 8 umliegenden Pixel definiert sind, in $\Gamma = \Gamma^0$ sinnvoll sein. Wir greifen jedoch im weiteren Verlauf ausschließlich auf eine Nachbarschaftsdefinition gemäß* (2.1) *zurück, da selbst für den Fall eines regulären Startgitters ab dem folgenden Zerlegungslevel der Konstruktion die Punkte $x_1^l, \ldots, x_{N/2^l}^l$ im Allgemeinen nicht mehr regulär verteilt sind.*

Beschreibt nun $(p^l(1), \ldots, p^l(k))$ den bisher konstruierten Pfad, so definieren wir die eingeschränkte Nachbarschaft

$$\tilde{N}_{C_1}(x_{p^l(k)}^l) := \{x_r^l \in N_{C_1}(x_{p^l(k)}^l) : r \notin \{p^l(1), \ldots, p^l(k)\}\}, \tag{2.2}$$

welche nur die Nachbarn enthält, die noch nicht auf dem Pfad gewählt wurden. Die Menge $\tilde{N}_{C_1}(x_{p^l(k)}^l)$ lässt sich weiter einschränken zu

$$\tilde{N}_{C_1,\theta}(x_{p^l(k)}^l) := \{x_r^l \in \tilde{N}_{C_1}(x_{p^l(k)}^l) : |c_{p^l(k)}^l - c_r^l| \leq \theta\}, \tag{2.3}$$

indem man die Punkte ausschließt, deren Tiefpass-Wert um mehr als eine Konstante $\theta > 0$ vom Tiefpass-Wert des aktuellen Punktes $x_{p^l(k)}^l$ abweicht. Weiterhin bezeichne

$$\tilde{N}_{k,l} := \{x_r^l \in \Gamma^l : r \notin \{p^l(1), \ldots, p^l(k)\}\} \tag{2.4}$$

die Menge aller verbleibenden, d.h. noch nicht auf dem Pfad gewählten, Punkte. Streicht man aus $\tilde{N}_{k,l}$ wiederum alle Punkte mit um mehr als θ abweichenden Tiefpass-Wert bezüglich des aktuellen Punktes $x_{p^l(k)}^l$, so erhält man die Menge

$$\tilde{N}_{k,l,\theta} := \{x_r^l \in \tilde{N}_{k,l} : |c_{p^l(k)}^l - c_r^l| \leq \theta\}. \tag{2.5}$$

Bemerkung 2.66 *Die Konstante θ kann levelabhängig gewählt werden, da sich der Wertebereich der Tiefpass-Werte c_r^l wegen $\sum_{k\in\mathbb{Z}} h_k = \sqrt{2}$ von Level zu Level um den Faktor $\sqrt{2}$ vergrößert. In der Praxis hat sich jedoch ein konstantes θ für alle Level als ausreichend erwiesen.*

Wir schlagen nun folgenden Algorithmus 2.67 zur Bestimmung eines Pfadvektors vor (siehe [66]).

Algorithmus 2.67 Adaptiv deterministische Pfadkonstruktion
Gegeben: $\Gamma^l = \{x_1^l, \ldots, x_{N/2^l}^l\} \subset \mathbb{R}^d,\ (c_j^l)_{j=1}^{N/2^l}$.

1. *Wähle den ersten Index des Pfades $p^l(1)$ zufällig aus $\{1, \ldots, N/2^l\}$ unter der Annahme, dass alle Indizes die gleiche Wahrscheinlichkeit besitzen, gewählt zu werden.*

2. *Für $k = 1, \ldots, N/2^l - 1$ durchlaufe folgende Schritte:*

   a) *Berechne $N_{C_1}(x_j^l), \tilde{N}_{C_1}(x_{p^l(k)}^l)$ und $\tilde{N}_{C_1,\theta}(x_{p^l(k)}^l)$ gemäß (2.1)-(2.3).*

   b) *Wähle den nächsten Index $p^l(k+1)$ des Pfades wie folgt:*

   - *Falls $\tilde{N}_{C_1,\theta}(x_{p^l(k)}^l)$ nicht die leere Menge ist, wähle $p^l(k+1)$, sodass*

$$x_{p^l(k+1)}^l = \underset{x \in \tilde{N}_{C_1,\theta}(x_{p^l(k)}^l)}{\operatorname{argmax}} \frac{\langle x_{p^l(k-1)}^l - x_{p^l(k)}^l, x_{p^l(k)}^l - x \rangle}{\|x_{p^l(k-1)}^l - x_{p^l(k)}^l\|_2 \cdot \|x_{p^l(k)}^l - x\|_2}, \tag{2.6}$$

     für $k > 1$. Für $k = 1$ wähle $p^l(k+1)$ zufällig unter den Indizes der Punkte in $\tilde{N}_{C_1,\theta}(x_{p^l(k)}^l)$ unter der Annahme einer Gleichverteilung.

   - *Falls $\tilde{N}_{C_1,\theta}(x_{p^l(k)}^l)$ leer ist, wähle $p^l(k+1)$ zufällig unter den Indizes der Punkte in $\tilde{N}_{C_1}(x_{p^l(k)}^l)$ unter der Annahme einer Gleichverteilung.*

   - *Falls $\tilde{N}_{C_1}(x_{p^l(k)}^l)$ leer ist, wähle $p^l(k+1)$ zufällig unter den Indizes der Punkte in $\tilde{N}_{k,l,\theta}$ (siehe (2.5)) unter der Annahme einer Gleichverteilung.*

   - *Falls $\tilde{N}_{k,l,\theta}$ leer ist, wähle $p^l(k+1)$ zufällig unter den Indizes der Punkte in $\tilde{N}_{k,l}$ (siehe (2.4)) unter der Annahme einer Gleichverteilung.*

Ausgabe: *Pfadvektor* $p^l = (p^l(k))_{k=1}^{N/2^l}$.

Bemerkung 2.68 *1. Man beachte, dass die adaptiv deterministische Pfadkonstruktion in Algorithmus 2.67 entgegen ihrer Benennung nicht vollständig deterministisch ist, sondern Zufallsanteile beinhaltet. Dies ist durchaus gewünscht, um für die Durchschnittsbildung in Algorithmus 2.63 verschiedene Pfade zu erhalten.*

2. *Eine effiziente Bestimmung der Nachbarschaftsmengen ist notwendig für den Algorithmus.*

3. *Sollte das Maximum von* (2.6) *in mehreren Punkten angenommen werden, so wähle man beliebig einen dieser Punkte als nächsten Pfadpunkt* $x^l_{p^l(k+1)}$.

4. *Die Wahl des nächsten Punktes* $x^l_{p^l(k+1)}$ *gemäß* (2.6) *minimiert den Winkel zwischen den Vektoren* $x^l_{p^l(k-1)} - x^l_{p^l(k)}$ *und* $x^l_{p^l(k)} - x^l_{p^l(k+1)}$, *sodass eine eingeschlagene Richtung des Pfades nach Möglichkeit beibehalten wird. Dies entspricht einer Verallgemeinerung des Prinzips der relaxierten EPWT (engl. relaxed EPWT) [101] in der Bildkompression.*

5. *Alternativ zu* (2.6) *kann man* $x^l_{p^l(k+1)}$ *für* $k > 1$ *derart wählen, dass*

$$x^l_{p^l(k+1)} = \operatorname*{argmin}_{x \in \tilde{N}_{C_1,\theta}(x^l_{p^l(k)})} |\|x^l_{p^l(k-1)} - x^l_{p^l(k)}\|_2 - \|x^l_{p^l(k)} - x\|_2|. \tag{2.7}$$

   Für $k = 1$ *setzen wir* $x_{p^l(0)} := x_{p^l(1)}$ *(d.h.* $\|x_{p^l(0)} - x_{p^l(1)}\|_2 = 0$ *) oder wir wählen* $p^l(k+1)$ *wieder zufällig unter den Indizes der Punkte in* $\tilde{N}_{C_1,\theta}(x^l_{p^l(k)})$ *unter der Annahme einer Gleichverteilung. In diesem Falle erhalten wir einen Pfad, auf dem die Abstände aufeinanderfolgender Paare von Punkten ähnlich sind, d.h. eine möglichst äquidistante Folge von Punkten. Eine Kombination der Bedingungen* (2.6) *und* (2.7) *ist denkbar.*

6. *Eine Konstruktion des Pfades unter Berücksichtigung von* (2.6) *bzw.* (2.7) *ist sinnvoll, da sich Eigenschaften der Wavelet-Filter - wie in Abschnitt 2.5 diskutiert - am besten für Funktionen, die äquidistant entlang von Geraden abgetastet wurden, ausnutzen lassen.*

7. *Anstatt den nächsten Punkt* $x^l_{p^l(k+1)}$, *wie im zweiten, dritten und vierten Fall in Algorithmus 2.67 beschrieben, zufällig gemäß einer Gleichverteilung aus der jeweiligen Kandidatenmenge zu wählen, ist auch eine Wahl, sodass* (2.6) *maximal,* (2.7) *minimal,* $|c^l_{p^l(k)} - c^l_{p^l(k-1)}|$ *minimal oder* $\|x^l_{p^l(k)} - x^l_{p^l(k-1)}\|_2$ *minimal wird, denkbar, jedoch mit höherem Aufwand verbunden.*

8. *In der Regel wird die Konstruktion gemäß Algorithmus 2.67 keinen vollständig durchgängigen Pfad ohne „Sprünge“ durch die Punktmenge* Γ^l *liefern, da bei einer Wahl des nächsten Punktes* $x^l_{p^l(k+1)}$ *gemäß der letzten zwei Fälle im zweiten Schritt* $x^l_{p^l(k+1)} \in N_{C_1}(x^l_{p^l(k)})$ *nicht mehr gegeben ist. Der nächste Punkt* $x^l_{p^l(k+1)}$ *liegt dann nicht mehr in der spatialen Nachbarschaft des aktuellen Punktes* $x^l_{p^l(k)}$. *Der Pfad macht gewissermaßen einen „Sprung“. In [101] und den Nachfolgearbeiten zur EPWT wird dies als „Unterbrechung“ (engl. interruption) des Pfades bezeichnet.*

2.4.2 Adaptiv zufällige Pfadkonstruktion

Die zweite von uns vorgeschlagene Konstruktion eines Pfades auf dem Datensatz beruht wesentlich stärker auf Zufall als die Konstruktion gemäß Algorithmus 2.67. Dies rechtfertigt die Namensgebungen „adaptiv *deterministisch*“ und „adaptiv *zufällig*“. Der nächste Punkt des Pfades soll nun stets zufällig unter den noch nicht gewählten Punkten bestimmt werden. Dabei versehen wir jedoch nicht jeden Punkt mit derselben Wahrscheinlichkeit. Punkte mit geringer spatialer Distanz zum aktuellen Pfadpunkt und geringer Differenz der zugeordneten Funktionswerte erhalten hohe Wahrscheinlichkeiten und werden somit bevorzugt als nächster Punkt des Pfades gewählt.

Wir betrachten die Datenpunkte des l-ten Level

$$y_j^l = ((x_j^l)^T, c_j^l)^T \in \mathbb{R}^{d+1}$$

für $j = 1, \ldots, N/2^l$ und erstellen eine symmetrische Gewichtsmatrix

$$W^l = (w(y_i^l, y_j^l))_{i,j=1}^{N/2^l}$$

mit Gewichten der Form

$$w(y_i^l, y_j^l) = w_1(x_i^l, x_j^l)\ w_2(c_i^l, c_j^l).$$

Für die Teilgewichte, d.h. für die Faktoren $w_1(x_i^l, x_j^l)$ und $w_2(c_i^l, c_j^l)$, sind unterschiedliche Ansätze denkbar. Wir verwenden in der Regel exponentielle Gewichtsfunktionen und setzen

$$w(y_i^l, y_j^l) = \exp\left(\frac{-\|x_i^l - x_j^l\|_2^2}{2^{2l/d}\eta_1}\right) \exp\left(\frac{-|c_i^l - c_j^l|^2}{2^l \eta_2}\right) \tag{2.8}$$

mit Parametern $\eta_1, \eta_2 > 0$. Derartige Gewichtsfunktionen werden u.a. bei den bilateralen Filtern in der Bildentstörung [132] oder auch im Clustering und in der Dimensionsreduktion (vgl. [29]) betrachtet. Man beachte auch die Ausführungen über die Laplaceschen Eigenabbildungen und die Diffusionsabbildungen in Kapitel 1 dieser Arbeit.

Die Normalisierung mit $\frac{1}{2^{2l/d}}$ in der Definition von w_1 erfolgt, da sich die Anzahl der Datenpunkte von Level zu Level halbiert und sich die paarweisen Abstände somit vergrößern. Die Definition von w_2 enthält die Normalisierung mit $\frac{1}{2^l}$, da die Größe des Wertebereichs der Tiefpass-Koeffizienten unter der Verwendung eines Tiefpass-Filters, der $\sum_{k\in\mathbb{Z}} h_k = \sqrt{2}$ erfüllt, von Level zu Level um den Faktor $\sqrt{2}$ wächst.

Aus Gründen der numerischen Effizienz greifen wir nicht auf die vollbesetzte Gewichtsmatrix W^l zurück, welche aus einer Festsetzung der Gewichte gemäß (2.8) hervorgeht. Stattdessen schneiden wir die Gewichte ab einer gewissen spatialen Distanz ab, indem wir mit einer Konstante $D_1 > 0$

$$w_1(x_i^l, x_j^l) = \begin{cases} \exp\left(-\frac{\|x_i^l - x_j^l\|_2^2}{2^{2l/d}\eta_1}\right) & \text{für } \|x_i^l - x_j^l\|_2 \le 2^{l/d} D_1 \\ 0 & \text{für } \|x_i^l - x_j^l\|_2 > 2^{l/d} D_1 \end{cases} \tag{2.9}$$

definieren. Auf diese Weise erhält man eine dünnbesetzte Gewichtsmatrix W^l. Alternativ kann man die Gewichte ab einer gewissen Differenz der zugeordneten Tiefpass-Werte durch entsprechende Definition von w_2 abschneiden.

Wir schlagen nun folgenden Algorithmus 2.69 zur adaptiv zufälligen Pfadkonstruktion vor (siehe [66]).

Algorithmus 2.69 Adaptiv zufällige Pfadkonstruktion
Gegeben: $y_j^l = ((x_j^l)^T, c_j^l)^T$ *für* $j = 1, \dots, N/2^l$.

1. *Berechne die Gewichtsmatrix* $W^l = (w(y_i^l, y_j^l))_{i,j=1}^{N/2^l}$.

2. *Wähle den ersten Index des Pfades* $p^l(1)$ *zufällig aus* $\{1, \dots, N/2^l\}$ *gemäß einer Gleichverteilung.*

3. *Durchlaufe folgende Iteration für* $k = 1, \dots, N/2^l - 1$*:*
   Nach der Konstruktion von $p^l(1), \dots, p^l(k)$ *im Pfadvektor betrachten wir die Teilmatrix* W_{k-1}^l*, welche man durch Streichung der* $p^l(1)$*-ten,*$\dots$*,*$p^l(k-1)$*-ten Zeilen und Spalten von* W^l *erhält, und berechnen*

$$s_{p^l(k)} = \sum_{\substack{j=1 \\ j \notin \{p^l(1), \dots, p^l(k)\}}}^{N/2^l} w(y_{p^l(k)}^l, y_j^l).$$

   Bestimme die Übergangswahrscheinlichkeiten $P_{p^l(k),r}$ *vom aktuellen Index* $p^l(k)$ *zu Index* r *vermöge*

$$P_{p^l(k),r} := \begin{cases} \frac{w(y_{p^l(k)}^l, y_r^l)}{s_{p^l(k)}} & \text{falls } r \in \{1, \dots, N/2^l\} \setminus \{p^l(1), \dots, p^l(k)\} \text{ und } s_{p^l(k)} \neq 0 \\ 0 & \text{sonst} \end{cases}$$

   und wähle $p^l(k+1)$ *zufällig gemäß der eingeführten Wahrscheinlichkeitsverteilung.*

Ausgabe: *Pfadvektor* $p^l = (p^l(k))_{k=1}^{N/2^l}$.

Bemerkung 2.70 *1. Arbeitet man mit abgeschnittenen Gewichten wie in* (2.9)*, so kann die Situation auftreten, dass alle Übergangswahrscheinlichkeiten* $P_{p^l(k),r}$ *verschwinden - nämlich dann, wenn alle entsprechenden spatialen Nachbarn des aktuellen Pfadpunktes bereits besucht wurden. In diesem Fall wählen wir den nächsten Pfadpunkt zufällig gemäß einer Gleichverteilung unter allen noch nicht besuchten Punkten, d.h. den Punkten* x_r^l *mit* $r \in \{1, \dots, N/2^l\} \setminus \{p^l(1), \dots, p^l(k)\}$*. In diesem Szenario müssen wir den Wunsch nach spatialer Nähe aufeinanderfolgender Punkte aufgeben und wie auch bei der adaptiv deterministischen Pfadkonstruktion (siehe 8. in Bemerkung 2.68) „Sprünge“ des Pfades erlauben.*

2. *Die Definition der Gewichte nach* (2.8) *bzw.* (2.9) *liefert hohe Übergangswahrscheinlichkeiten zwischen Punkten, die sich sowohl bezüglich ihrer spatialen Distanz als auch bezüglich der absoluten Differenz der zugeordneten Tiefpass-Werte nah sind. Die adaptiv zufällige Pfadkonstruktion fördert somit ein ähnliches Aussehen der Pfade wie die adaptiv deterministische Pfadkonstruktion (Algorithmus 2.67), bei der dies über die Definition der eingeschränkten Nachbarschaft gemäß* (2.3) *erreicht wird. Gegenüber der adaptiv deterministischen Pfadkonstruktion fehlt der adaptiv zufälligen Pfadkonstruktion jedoch die Bevorzugung der Einhaltung einer eingeschlagenen Richtung des Pfades (Bedingung* (2.6)*) oder die Bevorzugung von Äquidistanz der Pfadpunkte (Bedingung* (2.7)*).*

3. *Man beachte, dass sowohl die adaptiv deterministische als auch die adaptiv zufällige Pfadkonstruktion die gegebenen verrauschten Funktionswerte* $c_j^0 = \tilde{f}(x_j)$ *bzw. in höheren Leveln die Tiefpass-Werte* c_j^l *einbeziehen. Die Pfadkonstruktionen sind somit adaptiv. Nicht adaptive Pfadkonstruktionen, die nur auf die spatialen Koordinaten der Punkte zurückgreifen, sind denkbar (siehe Unterabschnitt 2.8.2). In diesem Falle ist eine Vorberechnung der Pfade möglich, was den rechnerischen Aufwand stark verringert. Die Entstörungsergebnisse sind jedoch deutlich weniger überzeugend.*

4. *In den Arbeiten zur EPWT, welche für Zwecke der Approximation und Komprimierung konzipiert wurde, wird viel Aufmerksamkeit auf die Verringerung der Adaptivitätskosten durch Hybridverfahren [106, 107], die relaxierte EPWT [101] sowie eine möglichst günstige Speicherung der Pfadvektoren (vgl. [88]) gelegt. Für die hier verfolgten Entstörungszwecke spielen die Speicherkosten der Pfadvektoren hingegen keine wesentliche Rolle.*

2.5 Eigenschaften der Wavelet-Transformation entlang von Pfaden

Der Erfolg von Wavelet-Shrinkage-Methoden, die das vorliegende Signal in Tiefpass- und Hochpass-Koeffizienten separieren und anschließend einen Shrinkage-Operator auf letztere anwenden, für Zwecke der Kompression und Entstörung beruht auf gewissen theoretischen Eigenschaften von Wavelet-Filtern, nämlich

- Reproduktion von Polynomen,
- verschwindende Momente,
- Abklingverhalten der Hochpass-Koeffizienten.

Wir untersuchen nun kurz, inwieweit sich diese klassischen Eigenschaften auf den Fall der gemäß Algorithmus 2.63 vorgeschlagenen Wavelet-Transformation entlang von Pfaden für gestreute Punkte $x_1, \ldots, x_N \in \mathbb{R}^d$ übertragen lassen (vgl. [66]).

Reproduktion von Polynomen

Wir können folgenden Satz (siehe [66]) bezüglich der Reproduktion von Polynomen festhalten.

Satz 2.71 *Sei $h = (h_k)_{k\in\mathbb{Z}}$ der Tiefpass-Filter des Zerlegungsteiles einer Wavelet-Filterbank perfekter Rekonstruktion mit $\sum_{k\in\mathbb{Z}} h_k = 1$. Weiterhin sei $p^0 = (p^0(j))_{j=1}^N$ der Pfadvektor zur Umordnung der Datenpunkte. Wir betrachten ein d-variates lineares Polynom $f(x) = a^T x + b$ mit $a \in \mathbb{R}^d$ und $b \in \mathbb{R}$. Dann reproduziert die Folge der Tiefpass-Koeffizienten $(c_j^1)_{j=1}^{N/2}$, welche man durch Anwendung von $(\bar{h}_{-k})_{k\in\mathbb{Z}}$ auf die Folge $(f(x_{p^0(j)}))_{j=1}^N$ erhält, das Polynom f in den Punkten $x_j^1 := \sum_{k\in\mathbb{Z}} \bar{h}_{k-2j+1} x_{p^0(k)}$, d.h.*

$$f(x_j^1) = c_j^1$$

für $j = 1, \ldots, N/2$.

Beweis. Wir wenden den Filter $(\bar{h}_{-k})_{k\in\mathbb{Z}}$ auf die Folge der Funktionswerte entlang des Pfades p^0 an, was uns

$$\begin{aligned} c_j^1 &= \sum_{k\in\mathbb{Z}} \bar{h}_{k-2j+1} f(x_{p^0(k)}) \\ &= \sum_{k\in\mathbb{Z}} \bar{h}_{k-2j+1}(a^T x_{p^0(k)} + b) = a^T\Big(\sum_{k\in\mathbb{Z}} \bar{h}_{k-2j+1} x_{p^0(k)}\Big) + b \sum_{k\in\mathbb{Z}} \bar{h}_{k-2j+1} \\ &= a^T\Big(\sum_{k\in\mathbb{Z}} \bar{h}_{k-2j+1} x_{p^0(k)}\Big) + b = a^T x_j^1 + b = f(x_j^1) \end{aligned}$$

liefert. ■

Im Gegensatz zur üblichen Wavelet-Transformation kann dieses Ergebnis nicht auf Polynome höherer Ordnung verallgemeinert werden. Limitierend ist hier, dass die konstruierten Pfade in spatialer Hinsicht stückweise linear sind.

Verschwindende Momente

Wir diskutieren die Eigenschaft verschwindender Momente. Man nehme an, dass der Wavelet-Filter $g = (g_k)_{k\in\mathbb{Z}}$ des Zerlegungsteiles der Wavelet-Filterbank den Bedingungen

$$\sum_{k\in\mathbb{Z}} g_k = 0$$

und

$$\sum_{k\in\mathbb{Z}} k g_k = 0$$

genügt. Betrachten wir eine konstante Funktion $f(x) = c$ mit $c \in \mathbb{R}$, so folgt für die Wavelet-Koeffizienten d_j^1, welche wir durch die Anwendung von $(\bar{g}_{-k})_{k\in\mathbb{Z}}$ entlang des Pfadvektors p^0 und Downsampling erhalten

$$d_j^1 = \sum_{k\in\mathbb{Z}} \bar{g}_{k-2j+1} f(x_{p^0(k)}) = c \sum_{k\in\mathbb{Z}} \bar{g}_{k-2j+1} = 0.$$

Für ein lineares Polynom $f(x) = a^T x + b$ mit $a \in \mathbb{R}^d$ and $b \in \mathbb{R}$ gilt

$$d_j^1 = a^T \Big(\sum_{k\in\mathbb{Z}} \bar{g}_{k-2j+1} x_{p^0(k)}\Big) + b \sum_{k\in\mathbb{Z}} \bar{g}_{k-2j+1} = a^T \Big(\sum_{k\in\mathbb{Z}} \bar{g}_{k-2j+1} x_{p^0(k)}\Big).$$

Die Wavelet-Koeffizienten d_j^1 verschwinden in diesem Fall demnach, wenn die Folge $(x_{p^0(k)})_{k\in\mathbb{Z}}$ kollinear und äquidistant ist. Die Pfadvektoren p^0 sollten folglich lokal so gewählt werden, dass die Punkte $x_{p^0(k)}$ äquidistant auf einer Geraden liegen. Die Bedingungen (2.6) bzw. (2.7) in Algorithmus 2.67 sollen dieses Verhalten des Pfades gewährleisten.

Abklingverhalten der Hochpass-Koeffizienten

Das Abklingverhalten der Hochpass-Koeffizienten der Wavelet-Transformation entlang von Pfaden ist aufwändig zu untersuchen, insbesondere da die Konstrukion nach Algorithmus 2.67 oder Algorithmus 2.69 einen Zufallsanteil aufweist. Wir können jedoch die in [105] von Plonka et al. im bivariaten Fall für die EPWT bewiesenen Resultate als Indiz für ein schnelles Abklingen der Hochpass-Koeffizienten für gewisse stückweise glatte Funktionen nehmen.

Man betrachte eine endliche Menge von paarweise disjunkten Teilmengen $\{\Omega_i\}_{1\le i\le K}$ von $[0,1)^2$, für die $\Omega := \bigcup_{i=1}^K \Omega_i = [0,1)^2$ gilt. Des Weiteren sei $f : \Omega \to \mathbb{R}$ eine bivariate Funktion, welche auf jeder Menge Ω_i eine Hölder-Bedingung erfüllt, d.h.

$$|f(x) - f(\tilde{x})| \le C\|x - \tilde{x}\|_2^\alpha \qquad \text{für alle } x, \tilde{x} \in \Omega_i \tag{2.10}$$

mit von i unabhängigen Konstanten $\alpha \in (0, 1]$ und $C > 0$. Diese Funktion f sei gleichmäßig abgetastet in Ω, sodass die Funktionswerte $f(\frac{j_1}{2^J}, \frac{j_2}{2^J})$ für $j_1, j_2 = 0, \dots 2^J - 1$, wobei $N = 2^{2J}$ ist, gegeben sind. Derartige stückweise Hölder-stetige Bilder werden auch als „Cartoon-ähnlich“ bezeichnet.

In [105] weisen Plonka et al. nach, dass die resultierenden EPWT-Wavelet-Koeffizienten d_k^l des l-ten Level der Abschätzung

$$|d_k^l| \leq \frac{1}{2} C_2^\alpha 2^{-(2J-l)(\alpha+1)/2}$$

mit einer Konstanten $C_2 > 0$ genügen, wenn die sogenannte Durchmesserbedingung (engl. diameter condition) und die sogenannte Regionenbedingung (engl. region condition) erfüllt sind. Zusätzlich liefert die EPWT nach [105] unter diesen Voraussetzungen eine asymptotisch optimale M-Term-Approximation f_M von f (die Approximation von f mit nur den M betragsmäßig größten EPWT-Wavelet-Koeffizienten). Denn mit einer Konstanten $C_3 > 0$ gilt

$$\|f - f_M\|_{L^2(\Omega)}^2 \lesssim C_3 N^{-\alpha}.$$

Die Durchmesserbedingung (vgl. [105]) misst gewissermaßen, wie gleichmäßig verteilt die Menge der gestreuten Punkte in Level l ist. Die Regionenbedingung (vgl. [105]) fordert, dass, wenn $x_{p^l(k)}^l \in \Omega_i$ ist, nach Möglichkeit auch der nächste Punkt $x_{p^l(k+1)}^l$ des Pfades in Ω_i liegt. Der Pfad soll also zunächst alle Punkte innerhalb eines Glattheitsgebietes ablaufen, bevor er in ein anderes übergeht.

Auch wenn wir für unsere neue Entstörungsmethode die Durchmesser- und Regionenbedingung nicht garantieren können, werden sie in den von uns verwendeten Pfadkonstruktionen gemäß Algorithmus 2.67 und Algorithmus 2.69 berücksichtigt. Bei der adaptiv deterministischen Pfadkonstruktion spiegelt sich die Durchmesserbedingung in der Definition der spatialen Nachbarschaften $\tilde{N}_{C_1}(x_{p^l(k)}^l)$ (siehe (2.2)) und die Regionenbedingung in der Definition der eingeschränkten Nachbarschaften $\tilde{N}_{C_1,\theta}(x_{p^l(k)}^l)$ (siehe (2.3)) wider. Bei der adaptiv zufälligen Pfadkonstruktion wird die Erfüllung der beiden Bedingungen gefördert, da wir Punkte mit geringem spatialen Abstand $\|x_{p^l(k)}^l - x_{p^l(k+1)}^l\|_2$ und geringer Differenz $|c_{p^l(k)}^l - c_{p^l(k+1)}^l|$ der zugeordneten Tiefpass-Werte mit höherer Wahrscheinlichkeit als nächsten Punkt des Pfades wählen.

Bemerkung 2.72 *In [102] erfolgt unter zusätzlichen Annahmen eine Verallgemeinerung der Ergebnisse aus [105] auf den Fall stückweise Hölder-glatter bivariater Funktionen mit beliebigen Hölder-Exponenten $\alpha > 0$, wobei die Bedingung* (2.10) *durch*

$$|f^{(\beta)}(x) - f^{(\beta)}(\tilde{x})| \leq C\|x - \tilde{x}\|_2^{\alpha-|\beta|} \qquad \textit{für alle } x, \tilde{x} \in \Omega_i$$

für alle Ableitungen $f^{(\beta)}$ mit $|\beta| = \lfloor \alpha \rfloor$ ersetzt wird.

2.6 Implementierung des Algorithmus 2.63

Dieser Abschnitt beschreibt, wie wir Algorithmus 2.63 in MATLAB implementiert haben (siehe [3]).

Hauptprogramm

Wir gehen von einer $N \times (d+2)$-Matrix $\mathfrak{D}$ aus, welche die N Datenpunkte mit ihren zugehörigen gestörten Funktionswerten darstellt. Die erste Spalte enthält die Indizes $1, \ldots, N$ der Datenpunkte. In den folgenden d Spalten sind die Koordinaten der Punkte $x_1, \ldots, x_N$ eingetragen. Die gestörten Funktionswerte $\tilde{f}(x_1), \ldots, \tilde{f}(x_N)$ stehen in der letzten Spalte. Die Matrix $\mathfrak{D}$ wird im Folgenden schrittweise manipuliert. Wir speichern die ersten $d+1$ Spalten der ursprünglichen Matrix $\mathfrak{D}$ separat ab, da wir sie am Ende des Algorithmus benötigen.

Zerlegung Wir beginnen mit der Pfadkonstruktion, auf deren Implementierung wir später noch eingehen, und erhalten den Pfadvektor, d.h. eine Permutation der Indizes $1, \ldots, N$. Wir sortieren die Zeilen der Matrix $\mathfrak{D}$ gemäß dieser Permutation und speichern den Pfadvektor im Hintergrund. Anschließend wenden wir die jeweiligen eindimensionalen Wavelet-Zerlegungsfilter mit nachfolgendem Downsampling auf die Vektoren, die die zweite bis letzte Spalte der umsortierten Matrix darstellen, an. Hierzu verwenden wir die Implementierung der Wavelet-Transformation von Getreuer [54]. Wir löschen jede zweite Zeile der Matrix (Downsampling) und überschreiben anschließend die zweite bis letzte Spalte mit den erhaltenen Tiefpass-Werten nach Downsampling (Aktualisierung der Punkte bzw. Funktionswerte). Die erhaltenen Hochpass-Werte für die letzte Spalte werden nach Anwendung des Shrinkage-Operators abgespeichert. Die aus der zweiten bis vorletzten Spalte (Koordinaten der Punkte) resultierenden Hochpass-Werte werden nicht benötigt und müssen daher nicht berechnet werden.

Das erste Zerlegungslevel ergibt also eine modifizierte $N/2 \times (d+2)$-Matrix sowie die abgespeicherten Hochpass-Werte und den abgespeicherten Pfadvektor. Wir iterieren das obige Vorgehen mit der modifizierten Matrix für weitere Zerlegungslevel.

Rekonstruktion Die Rekonstruktion der Funktionswerte verläuft ebenso levelweise iterativ. Wir initialisieren einen Vektor c als letzte Spalte der zuvor nach dem letzten Zerlegungslevel erhaltenen Matrix. Der Vektor c enthält folglich die Tiefpass-Werte des letzten Zerlegungslevel. Wir fügen die entsprechend separat gespeicherten Hochpass-Werte zu c hinzu und wenden nach vorherigem Upsampling die entsprechenden Wavelet-Rekonstruktionsfilter an, um die geglätteten Tiefpass-Werte des nächsten Level zu er-

halten. Diese sortieren wir anschließend mittels der durch den gespeicherten Pfadvektor induzierten inversen Permutation des zugehörigen Level um. Man beachte, dass eine Rekonstruktion der spatialen Koordinaten der Punkte nicht notwendig ist.

Wir wiederholen dieses Vorgehen, bis wir im „nullten“ Level anlangen, sodass c die geglätteten Funktionswerte darstellt. Abschließend fügen wir den Vektor c und die zu Beginn gespeicherten ersten $d+1$ Spalten der originalen Matrix $\mathfrak{D}$ zu einer Matrix zusammen, die die N Datenpunkte mit ihren zugehörigen entstörten Funktionswerten beschreibt.

Adaptiv deterministische Pfadkonstruktion

Wir beschreiben kurz die Grundlagen der Implementierung der Pfadkonstruktionen. Wir setzen voraus, dass wir die (spatialen) Nachbarschaften der Datenpunkte vorab berechnet haben. Auf diesen Aspekt gehen wir im Anschluss noch ein.

Wir versehen den Vektor $(x_j^l)_{j=1}^{N/2^l}$ der $N/2^l$ Punkte mit einem Label 1 bzw. 0 an der Stelle j, wenn der betreffende Punkt x_j^l bei der Pfadkonstruktion noch nicht besucht bzw. bereits besucht wurde.

Der Pfadvektor, der aus den Indizes der Pfadpunkte in der Reihenfolge ihres Besuches besteht, wird rekursiv konstruiert. Die sukzessive Einschränkung der betrachteten Nachbarschaften wird über logische Indizierung geregelt.

Adaptiv zufällige Pfadkonstruktion

Unsere Implementierung der adaptiv zufälligen Pfadkonstruktion hat die gleiche Grundstruktur wie die Implementierung der adaptiv deterministischen Pfadkonstruktion. Man beachte hierzu, dass sich das „Abschneiden“ der Wahrscheinlichkeiten gemäß (2.9) auch derart interpretieren lässt, dass wir als nächsten Pfadpunkt nur entsprechende spatiale Nachbarn zulassen.

Als Zufallsgenerator nutzen wir die interne MATLAB-Funktion „rand“, welche eine Pseudo-Zufallszahl ζ zwischen 0 und 1 gemäß einer Gleichverteilung liefert. Sind nun $x_{p^l(k)}^l$ der aktuelle Pfadpunkt und $x_{r_i}^l$ für $i = 1, \ldots, R$ mit $R \leq N/2^l$ die Kandidaten für den nächsten Pfadpunkt $x_{p^l(k+1)}^l$ mit zugehörigen Wahrscheinlichkeiten $P_{p^l(k),r_i}$, so betrachten wir die Intervalle $I_{r_i} = [F_{r_i-1}, F_{r_i})$ mit

$$F_{r_i} = \begin{cases} 0 & \text{für } i = 0 \\ \sum_{j=1}^{i} P_{p^l(k),r_j} & \text{sonst} \end{cases}.$$

Wir wählen $x_{r_i}^l$ genau dann als nächsten Pfadpunkt, wenn die Zufallszahl ζ im Intervall I_{r_i} liegt.

Berechnung der Nachbarschaften

Wir diskutieren im Folgenden die Berechnung der Nachbarschaften

$$N_{C_1}(x_i^l) := \{x_k^l \in \Gamma^l : \|x_i^l - x_k^l\|_2 \leq 2^{l/d} C_1, i \neq k\},$$

welche den größten Aufwand bei der Pfadkonstruktion darstellt. Zu beachten ist, dass wir die Nachbarschaften $N_{C_1}(x_i^l)$ für jeden Punkt x_i^l für $i = 1, \ldots, N/2^l$ benötigen. Bei einem naiven Ansatz würden viele Abgleiche paarweiser Distanzen daher mehrfach erfolgen. Wir greifen auf eine Möglichkeit für ein effizienteres Vorgehen zurück. Dabei sei im Folgenden

$$\varepsilon := 2^{l/d} C_1.$$

Wir nutzen zwei Ideen, welche von meinem Kollegen Florian Boßmann stammen, um die Nachbarschaftsbestimmung effizienter zu machen. Ein ähnlicher Ansatz findet sich auch in [78]. Die erste Idee verwendet den Schätzer

$$d_\triangle(i,k) := |\|x_i^l\|_2 - \|x_k^l\|_2|$$

für die euklidischen Abstände

$$d(i,k) := \|x_i^l - x_k^l\|_2,$$

welcher sich aus der umgekehrten Dreiecksungleichung

$$|\|x_i^l\|_2 - \|x_k^l\|_2| \leq \|x_i^l - x_k^l\|_2$$

ergibt. Dabei lassen sich die Werte $d_\triangle(i,k)$ relativ schnell bestimmen, wenn wir die quadrierten euklidischen Normen

$$\rho(k) := \|x_k^l\|_2^2$$

aller Punkte x_k^l vorberechnen. Wir grenzen dann zunächst die Menge

$$\mathcal{N}_{\triangle,\varepsilon}(i) := \{k \in \{1, \ldots, N/2^l\} \setminus \{i\} : d_\triangle(i,k) \leq \varepsilon\}$$

ein, bevor wir unter diesen Kandidatenindizes die Menge der Indizes

$$\mathcal{N}_\varepsilon(i) := \{k \in \mathcal{N}_{\triangle,\varepsilon}(i) : d(i,k) \leq \varepsilon\}$$

der Nachbarschaft gemäß der exakten euklidischen Abstände $d(i,k) = (\rho(i) + \rho(k) - 2\langle x_i^l, x_k^l\rangle)^{1/2}$ bestimmen. Wir ersparen uns auf diese Weise eine Berechnung der euklidischen Abstände $d(i,k)$ für alle $i,k = 1, \ldots, N$.

Die zweite Idee nutzt aus, dass die ε-Nachbarschaft eines Punktes x_j^l in der 2ε-Nachbarschaft des Punktes x_i^l enthalten ist, wenn x_j^l in der ε-Nachbarschaft von x_i^l liegt. Wir bestimmen daher bei der Berechnung der ε-Nachbarschaft von x_i^l auch die 2ε-Nachbarschaft von x_i^l. Im direkten Anschluss untersuchen wir zunächst die Punkte der ε-Nachbarschaft von x_i^l auf ε-Nachbarn, wobei wir unsere Suche auf die 2ε-Nachbarschaft von x_i^l beschränken können. Auf diese Weise reduzieren wir die Anzahl der nötigen Abgleiche paarweiser Abstände gegen den Parameter ε.

Durch Kombination der beiden Ideen erhält man Algorithmus 2.73 [21] als Vorschlag für eine effiziente Berechnung der Nachbarschaften aller Punkte.

Algorithmus 2.73 Berechnung der Nachbarschaften
Eingabe: *Punkte x_i^l für $i = 1, \ldots, N/2^l$ und Nachbarschaftsgröße $\varepsilon = 2^{l/d} C_1$.*

- *Berechne $\rho(i) = \|x_i^l\|_2^2$ für alle $i = 1, \ldots, N/2^l$ und initialisiere die Menge der Indizes, deren Nachbarschaften bereits bestimmt wurden: $\mathcal{I} = \emptyset$.*

- *Für $i = 1, \ldots, N/2^l$:*
  Falls $i \in \mathcal{I}$ ist, fahre mit $i + 1$ fort. Andernfalls aktualisiere $\mathcal{I}$ durch Hinzufügung von i, d.h. $\mathcal{I} \mapsto \mathcal{I} \cup \{i\}$, und führe die folgenden Schritte durch.

  1. *Bestimme die Menge $\mathcal{N}_{\triangle,2\varepsilon}(i) = \{k \in \{1, \ldots, N/2^l\} \setminus \{i\} : d_\triangle(i,k) \leq 2\varepsilon\}$, um die Kandidaten für die Indizes der 2ε-Nachbarschaft von x_i^l durch Verwendung des Schätzers $d_\triangle$ einzugrenzen.*

  2. *Berechne die exakten euklidischen Abstände $d(i,k) = (\rho(i)+\rho(k)-2\langle x_i^l, x_k^l\rangle)^{1/2}$ für $k \in \mathcal{N}_{\triangle,2\varepsilon}(i)$ und bestimme durch Suche in $\mathcal{N}_{\triangle,2\varepsilon}(i)$ die Indizes der 2ε-Nachbarschaft $\mathcal{N}_{2\varepsilon}(i) = \{k \in \mathcal{N}_{\triangle,2\varepsilon}(i) : d(i,k) \leq 2\varepsilon\}$.*

  3. *Bestimme die Indizes der ε-Nachbarschaft $\mathcal{N}_\varepsilon(i) = \{k \in \mathcal{N}_{2\varepsilon}(i) : d(i,k) \leq \varepsilon\}$ durch Suche in $\mathcal{N}_{2\varepsilon}(i)$.*

  4. *Setze $\mathcal{J} = \mathcal{N}_\varepsilon(i) \setminus \mathcal{I}$ und durchlaufe für $j \in \mathcal{J}$ die folgenden Schritte:*

     a) *Füge j zur Menge der betrachteten Indizes hinzu, d.h. $\mathcal{I} \mapsto \mathcal{I} \cup \{j\}$.*

     b) *Berechne $d(j,k)$ für $k \in \mathcal{N}_{2\varepsilon}(i)$ und bestimme die Indexmenge $\mathcal{N}_\varepsilon(j) = \{k \in \mathcal{N}_{2\varepsilon}(i) : d(j,k) \leq \varepsilon\}$ durch Suche in $\mathcal{N}_{2\varepsilon}(i)$.*

Ausgabe: *Indexmengen $\mathcal{N}_\varepsilon(i)$ der Nachbarschaften der Punkte x_i^l für $i = 1, \ldots, N/2^l$.*

Man beachte, dass wir Algorithmus 2.73 für jede Punktmenge Γ^l und jede Iteration des Algorithmus 2.63, d.h. bei Betrachtung von T Iterationen und L Zerlegungsleveln TL-

mal, durchführen müssen. Hierbei werden die Nachbarschaften vieler Punkte mehrfach aufs Neue bestimmt. Es ergibt sich somit weiteres Einsparungspotenzial hinsichtlich der Kosten der Nachbarschaftsbestimmung. Dies betrifft insbesondere den in Bemerkung 2.64 beschriebenen Fall, in dem zur Aktualisierung der Punktmenge Γ^l ein reines Downsampling verwendet wird. In diesem Fall entstehen keine neuen Punkte in Γ^l, die nicht schon in $\Gamma = \Gamma^0$ enthalten sind.

2.7 Numerische Resultate

Dieser Abschnitt präsentiert Ergebnisse der Anwendung unserer MATLAB-Implementierung gemäß Abschnitt 2.6 der vorgestellten Entstörungsmethode nach Algorithmus 2.63 für verrauschte Bilder. Zur Einordnung der erreichten Resultate stellen wir auch die Entstörungsergebnisse mittels einiger vergleichbarer Methoden gegenüber. Diese Methoden sind in Tabelle 2.4 aufgelistet.

1.	Shrinkage der zweidimensionalen Tensorprodukt-Wavelet-Transformation ohne Cycle-Spinning [93]
2.	Shrinkage der zweidimensionalen Tensorprodukt-Wavelet-Transformation mit Cycle-Spinning (64 Verschiebungen) [31]
3.	Vier-Pixel-Schema [141]
4.	Curvelet-Shrinkage [124]
5.	Shearlet-Shrinkage [47]
6.	BM3D [36]
7.	Algorithmus 2.63 mit 64 Iterationen und adaptiv deterministischer Pfadkonstruktion
8.	Algorithmus 2.63 mit 64 Iterationen und adaptiv zufälliger Pfadkonstruktion

Tabelle 2.4: Verglichene Methoden.

Man beachte, dass die Methoden 1 und 3-5 aus Tabelle 2.4 keine mit dem Cycle-Spinning vergleichbare Durchschnittsbildung enthalten. Allerdings handelt es sich bei Curvelets und Shearlets um Systeme mit von Natur aus höherer Redundanz.

Wir weisen darauf hin, dass wir uns für den Vergleich auf Daten auf einem zweidimensionalen regulären Gitter (Bilder) beschränken. Die Entstörungsmethoden 1-6 aus Tabelle 2.4 sind in dieser Form nur für diese spezielle Klasse von Datensätzen anwendbar. Es existieren theoretische Erweiterungen der Curvelets (siehe z.B. [26]) und Shearlets (siehe z.B. [59]) auf beliebige Raumdimensionen sowie dreidimensionale Erweiterungen des BM3D-Algorithmus zur Entstörung von Videos (siehe z.B. [35]). Diese Erweiterungen benötigen jedoch ein reguläres Gitter.

Zur Erläuterung der Methoden 1 und 2 sei auf Abschnitt 2.2 und Algoritmus 2.59 verwiesen. Für die zweidimensionalen Wavelet-Transformationen verwenden wir erneut die Implementierung von Getreuer [54]. Dabei wählen wir - wie auch für die eindimensionale Wavelet-Transformation bei der Entstörung mittels Algorithmus 2.63 - biorthogonale CDF-9/7-Filter.

Wir beschreiben noch kurz die Grundideen der verbleibenden Methoden.

Vier-Pixel-Schema

Das Vier-Pixel-Schema (engl. Four-Pixel Scheme) [141] ist ein spezielles numerisches Schema für die Bildentstörung mittels partieller Differentialgleichungen. Filterung eines Bildes gemäß der iterativen Lösung einer diskretisierten partiellen Differentialgleichung, welche einen nicht linearen Diffusionsprozess mit dem verrauschten Bild als Anfangsbedingung beschreibt, stellt eine verbreitete Methode zur Bildentstörung dar. Dabei werden vermehrt unbeschränkte Diffusivitätsfunktionen mit einer Singularität im Nullpunkt betrachtet, was in Implementierungen resultiert, die unter langsamer Konvergenz oder numerischen Instabilitäten leiden. Diesem begegnet man häufig mit einer Regularisierung, die zu einer beschränkten Diffusivitätsfunktion führt. Durch diese Regularisierung verliert man einige der theoretischen Eigenschaften der unbeschränkten Diffusivitätsfunktionen und erhält oft Artefakte bei der Bildentstörung (vgl. [141]). Statt einer Regularisierung schlagen Welk et al. daher das Vier-Pixel-Schema vor, welches leicht zu implementieren und stabil ist. Das Schema beruht auf analytischen Lösungen der betrachteten partiellen Differentialgleichung für 2×2-Bilder, welche mittels einer Art additiven Operator-Splittings (AOS) kombiniert werden.

Curvelets und Shearlets

Wavelets können im Allgemeinen im zweidimensionalen Fall gerichtete Unstetigkeiten, etwa Unstetigkeiten entlang von Kurven, nicht optimal behandeln. Dies war eines der Hauptmotive für die Einführung der adaptiven EPWT zur Bildkompression. Bei der Curvelet-Transformation und der Shearlet-Transformation handelt es sich um nicht adaptive mit der Wavelet-Transformation verwandte Konzepte, die die Erfassung gerichteter Strukturen verbessern.

Bei den Curvelets [27] verwendet man Funktionensysteme

$$\{\phi_{j,k,l} : k \in \mathbb{Z}^2, l \in \{0, \dots, N_j - 1\}\}$$

mit

$$\phi_{j,k,l} := \phi_{j,0,0}(R_{\theta_{j,l}}(x - b_k^{j,l}))$$

für $N_j := 4 \cdot 2^{\lceil j/2 \rceil}$ und $j \in \mathbb{N}_0$. Dabei ist $R_{\theta_{j,l}}$ mit $\theta_{j,l} = \frac{\pi l 2^{-\lceil j/2 \rceil}}{2}$ eine Rotationsmatrix, $b_k^{j,l} = b_{(k_1,k_2)}^{j,l} = R_{\theta_{j,l}}^{-1} \left(\frac{k_1}{2^j}, \frac{k_2}{2^{j/2}}\right)^T$ eine Verschiebung und $\phi_{j,0,0}$ das sogenannte Basis-Curvelet nach Dilatation (engl. dilated basic curvelet), für dessen Fourier-Transformierte in Polarkoordinaten

$$\hat{\phi}_{j,0,0}(r,w) = 2^{-3j/4} W(2^{-j}r) \tilde{V}_{N_j}(w)$$

mit passenden Fensterfunktionen W und $\tilde{V}_{N_j}$ gilt. Man erhält das Funktionensystem $\{\phi_{j,k,l} : j \in \mathbb{N}_0, k \in \mathbb{Z}^2, l \in \{0, \ldots, N_j - 1\}\}$ aus einer Mutterfunktion ϕ folglich durch Dilatation und Translation wie bei den Wavelets und zusätzlich durch Rotation. Das System induziert ein sogenanntes Tiling des zweidimensionalen Frequenz-Raumes. Die Anzahl der Keile (engl. wedges) dieses Tilings in Skalierungslevel 2^{-j} beträgt dabei N_j (vgl. [87]).

Für die Shearlets [79] nutzt man zur Analyse den Shearlet-Frame

$$\{\psi_{i,j,k} : i, j \in \mathbb{Z}, k \in \mathbb{Z}^2\},$$

wobei

$$\psi_{i,j,k} := |\det A|^{i/2} \psi(B^j A^i \cdot - k)$$

mit

$$A := \begin{pmatrix} 2 & 0 \\ 0 & \sqrt{2} \end{pmatrix}$$

und

$$B := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

definiert ist. Man erhält die Frame-Elemente aus einer Mutterfunktion ψ also durch Translation und Dilatation (Matrix A) wie bei den Wavelets und zusätzlich durch Scherung (Matrix B). Man beachte weiterhin, dass die Dilatation im Gegensatz zu den Wavelets anisotrop ist.

Die Entstörung durch Curvelet-Shrinkage bzw. Shearlet-Shrinkage wurde mittels der Software CurveLab [1] bzw. ShearLab (Version 1.0) [4] getestet. Es sei darauf hingewiesen, dass in [47] mittels der Shearlet Transformation ohne Subsampling (engl. Nonsubsampled Shearlet Transform, NSST) bessere Entstörungsergebnisse als die hier später vorgestellten erreicht wurden.

BM3D

Bei der Bildentstörung mittels Abgleich von Blöcken und dreidimensionaler Filterung (engl. block-matching and 3D filtering, BM3D) [36] kombinieren Dabov et al. mehrere Ideen zu einem hervorragend funktionierendem Entstörungsschema.

Die Entstörung mit BM3D verläuft grundsätzlich in zwei Schritten. Der erste Schritt, gewissermaßen eine Vorentstörung, arbeitet wie folgt: Es werden überlappende Bildblöcke einer festgelegten Größe betrachtet. Man fixiert nacheinander je einen Referenzblock und sucht andere Blöcke mit hoher Korrelation zu diesem Referenzblock. Als Maß für die Korrelation dient hierbei der Frobenius-Abstand der z.B. mit einer zweidimensionalen diskreten Kosinus-Transformation mit anschließendem Hard-Shrinkage behandelten Blöcke. Dieses Korrelationsmaß ordnet Blöcken mit hoher Korrelation kleine positive Werte zu. Blöcke, für die das Korrelationsmaß einen Parameter $\tau_{\text{match}} > 0$ unterschreitet, werden folglich zu einem „Stapel“ zusammengefasst (engl. „stacked to form a 3D array“). Dieser Abgleich von Blöcken (engl. block-matching) liefert dreidimensionale „Stapel“, auf die man eine dreidimensionale unitäre Transformation mit Hard-Shrinkage zur Beseitigung des Rauschens anwendet. Durch anschließende Anwendung der inversen dreidimensionalen Transformation erhält man eine Rekonstruktion des dreidimensionalen „Stapels“, welche mit einem Gewicht in Abhängigkeit von der Varianz des „Stapels“ versehen wird. Dieses Vorgehen wird wiederholt, bis jeder Bildblock als fixierter Referenzblock fungiert hat. Letztlich setzt man in einem Aggregationsschritt die Blöcke wieder zu einem Bild zusammen. Dabei bildet man einen gewichteten Durchschnitt, der

- Überlappungen von Blöcken,
- Vorkommen von Blöcken in unterschiedlich vielen „Stapeln“ und
- die Gewichtungen der „Stapel“

berücksichtigt. Das Resultat ist ein (vor-)entstörtes Bild.

Der zweite Schritt des BM3D-Algorithmus funktioniert nach dem gleichen Schema wie der erste Schritt. Als Korrelationsmaß beim Abgleich der gestörten Blöcke dient diesmal jedoch der Frobenius-Abstand der um ihren Erwartungswert zentrierten Blöcke des aus dem ersten Schritt resultierenden vorentstörten Bildes. Weiterhin verwendet man zur Eliminierung des Rauschens mittels der dreidimensionalen Transformation der „Stapel“ statt des Hard-Shrinkage einen Wiener-Filter, der sich aus dem vorentstörten Bild ergibt. Das Resultat des zweiten Schrittes stellt dann die finale Rekonstruktion des Bildes dar.

Für eine äußerst effiziente Implementierung des BM3D-Algorithmus sei auf [2] verwiesen.

Resultate

Wir entstören exemplarisch die Grauwert-Bilder „Cameraman“ und „Peppers“ der Größe 256×256 Pixel mit synthetisch eingefügtem additiven Gaußschen Rauschen, wobei die Standardabweichung der zugehörigen Normalverteilung $\sigma = 0.1$ bzw. $\sigma = 0.15$ betrage (siehe Abbildung 2.5). Die Grauwerte seien im Intervall $[0, 1]$ kodiert. Wir haben

versucht, die jeweiligen freien Parameter der Entstörungsmethoden aus Tabelle 2.4 experimentell optimal zu bestimmen. Die für die hier präsentierten Ergebnisse verwendeten Parameter sind in Tabelle 2.5 aufgeführt. Dabei bezeichnet θ den jeweiligen Threshold-Parameter, τ die Schrittweite beim Vier-Pixel-Schema und C_1 bzw. D_1 den Nachbarschaftsparameter für unsere Pfadkonstruktion (siehe (2.2) bzw. (2.9)). Die Parameter η_1, η_2 werden für die Gewichte bei der adaptiv zufälligen Pfadkonstruktion (siehe (2.8)) benötigt.

Als Qualitätsmaß der Entstörung verwenden wir das gemäß

$$\mathrm{PSNR}(f, \tilde{f}) = 10 \log_{10} \frac{1}{\frac{1}{N} \sum_{j=1}^{N} (f(x_j) - \tilde{f}(x_j))^2}$$

in Dezibel (dB) definierte Spitzen-Signal-Rausch-Verhältnis (engl. Peak Signal-to-Noise Ratio, PSNR), wobei $f \in \mathbb{R}^N$ das rauschfreie Original und $\tilde{f} \in \mathbb{R}^N$ das verrauschte Bild seien. Die 1 im Zähler des Argumentes des Logarithmus stellt den quadrierten maximalen Bildwert dar.

Bemerkung 2.74 *Man beachte, dass der Zufall Einfluss auf das Ergebnis von Algorithmus 2.63 hat. Eine zweimalige Anwendung von Algorithmus 2.63 auf dasselbe verrauschte Bild wird somit im Allgemeinen zwei unterschiedliche Ergebnisse liefern. In der Praxis lagen die Abweichungen der erreichten PSNR-Werte dabei allerdings im Bereich von Hundertsteln eines Dezibels.*

Die Resultate der numerischen Experimente sind in den Abbildungen 2.6-2.13 und Tabelle 2.6 dargestellt. Man erkennt in allen vier Beispielen, dass die Entstörung mittels unseres Algorithmus 2.63 mit adaptiv deterministischer Pfadkonstruktion - mit Ausnahme des BM3D-Algorithmus, welcher den neuesten Stand der Technik repräsentiert - den besten PSNR-Wert liefert. Insbesondere liefert die adaptiv deterministische Pfadkonstruktion bessere Resultate als die adaptiv zufällige Pfadkonstruktion. Der Grund hierfür dürfte sein, dass wir bei der adaptiv zufälligen Pfadkonstruktion im Gegensatz zur adaptiv deterministischen Pfadkonstruktion die Beibehaltung eingeschlagener Richtungen des Pfades nicht fördern.

Abbildung 2.14 zeigt die Verteilung der aktualisierten Mengen Γ^l der gestreuten Punkte in unterschiedlichen Leveln exemplarisch für die Anwendung unseres Entstörungsalgorithmus mit der adapativ deterministischen Pfadkonstruktion auf das „Cameraman"-Bild. Man sieht, dass die für die Pfadkonstruktion des folgenden Level verwendeten Punktmengen (annähernd) quasi-uniform (siehe Definition 2.61) im ursprünglichen Definitionsbereich $[1, 256]^2$ sind, sodass der gesamte Bereich gut mit Informationen in Form der den Punkten $x_j^l \in \Gamma^l$ zugeordneten Tiefpass-Werte c_j^l abgedeckt ist. Weiterhin wird deutlich, dass Algorithmus 2.63 auch für nicht äquidistante Punkte funktioniert, denn

	„Peppers“, $\sigma = 0.1$	„Peppers“, $\sigma = 0.15$
Tensorprodukt-Wavelet	4 Level, $\theta = 0.29$	2 Level, $\theta = 0.55$
Tensorprodukt-Wavelet mit Cycle-Spinning	4 Level, $\theta = 0.29$	3 Level, $\theta = 0.45$
Vier-Pixel-Schema	76 Iterationen, $\tau = 0.001$	124 Iterationen, $\tau = 0.001$
Curvelet	$\theta = 0.31$	$\theta = 0.41$
Shearlet	$\theta = 0.001$	$\theta = 0.0015$
BM3D	Blockgröße 8×8, $\tau_{\text{match}} = 3000$ (1. Schritt) bzw. $\tau_{\text{match}} = 400$ (2. Schritt)	
Algorithmus 2.63, adaptiv deterministisch	6 Level, $C_1 = 1.3$, $\theta = 0.35$	6 Level, $C_1 = 1.1$, $\theta = 0.50$
Algorithmus 2.63, adaptiv zufällig	4 Level, $D_1 = 5$, $\theta = 0.28$, $\eta_1 = \eta_2 = 0.2$	4 Level, $D_1 = 5$, $\theta = 0.50$, $\eta_1 = \eta_2 = 0.2$
	„Cameraman“, $\sigma = 0.1$	„Cameraman“, $\sigma = 0.15$
Tensorprodukt-Wavelet	2 Level $\theta = 0.30$	3 Level, $\theta = 0.52$
Tensorprodukt-Wavelet mit Cycle-Spinning	4 Level $\theta = 0.30$	3 Level, $\theta = 0.45$
Vier-Pixel-Schema	73 Iterationen, $\tau = 0.001$	122 Iterationen, $\tau = 0.001$
Curvelet	$\theta = 0.31$	$\theta = 0.41$
Shearlet	$\theta = 0.001$	$\theta = 0.0015$
BM3D	Blockgröße 8×8, $\tau_{\text{match}} = 3000$ (1. Schritt) bzw. $\tau_{\text{match}} = 400$ (2. Schritt)	
Algorithmus 2.63, adaptiv deterministisch	6 Level, $C_1 = 1.3$, $\theta = 0.35$	8 Level, $C_1 = 1.4$, $\theta = 0.54$
Algorithmus 2.63, adaptiv zufällig	8 Level, $D_1 = 5$, $\theta = 0.30$, $\eta_1 = \eta_2 = 0.2$	8 Level, $D_1 = 5$, $\theta = 0.50$, $\eta_1 = \eta_2 = 0.2$

Tabelle 2.5: Verwendete Parameter für die Entstörung mittels der jeweiligen Methoden.

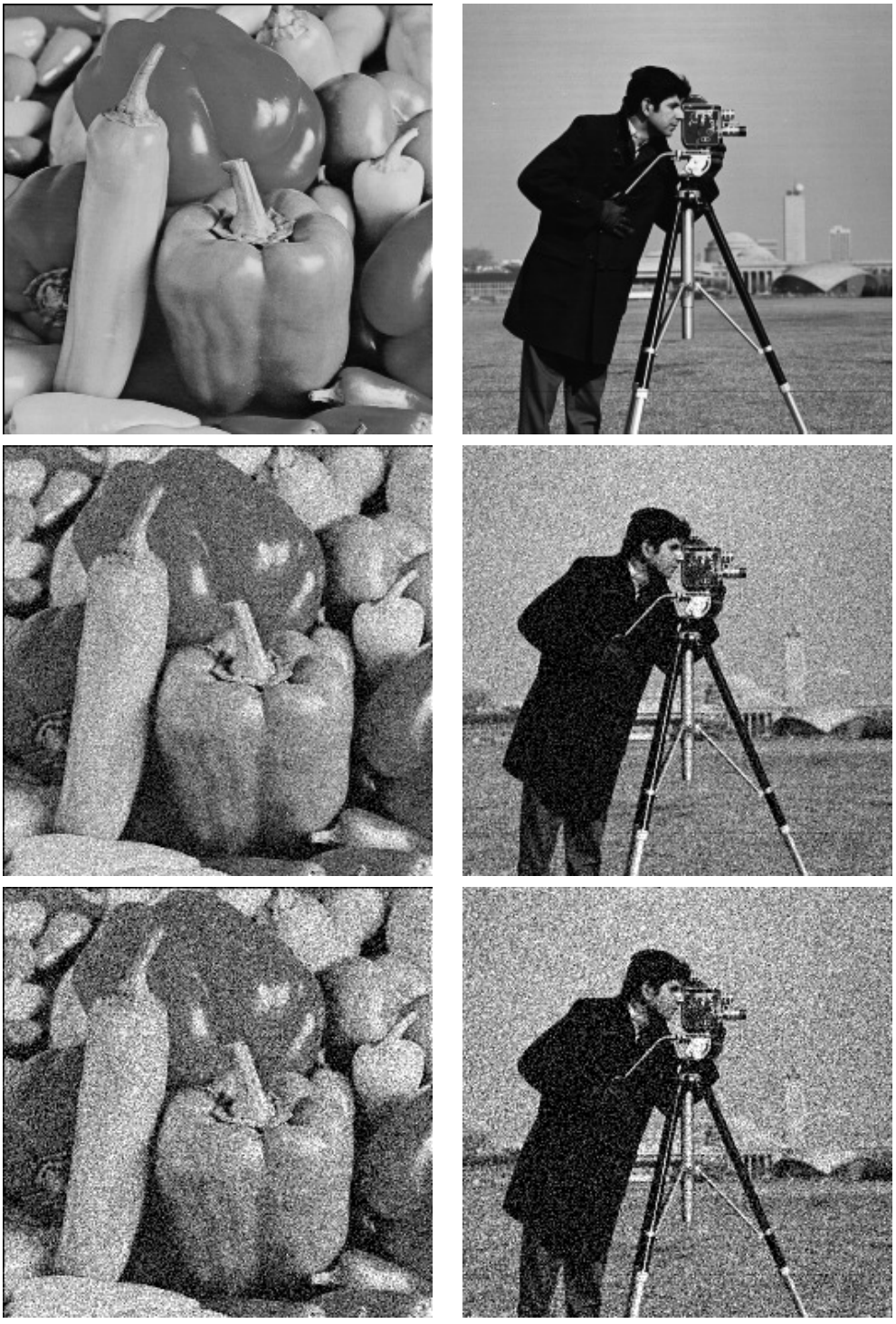

Abbildung 2.5: „Peppers“-Bild (links) und „Cameraman“-Bild (rechts); obere Reihe: Originale; mittlere Reihe: mit Gaußschem Rauschen ($\sigma = 0.1$) belegt; untere Reihe: mit Gaußschem Rauschen ($\sigma = 0.15$) belegt.

Abbildung 2.6: Entstörung des „Peppers“-Bildes ($\sigma = 0.1$) mittels Tensorprodukt-Wavelets ohne (oben links) und mit Cycle-Spinning (oben rechts), Vier-Pixel-Schema (unten links) und Curvelets (unten rechts).

Abbildung 2.7: Entstörung des „Peppers“-Bildes ($\sigma = 0.1$) mittels Shearlets (oben links), BM3D (oben rechts), Algorithmus 2.63 mit adaptiv deterministischer (unten links) und adaptiv zufälliger Pfadkonstruktion (unten rechts).

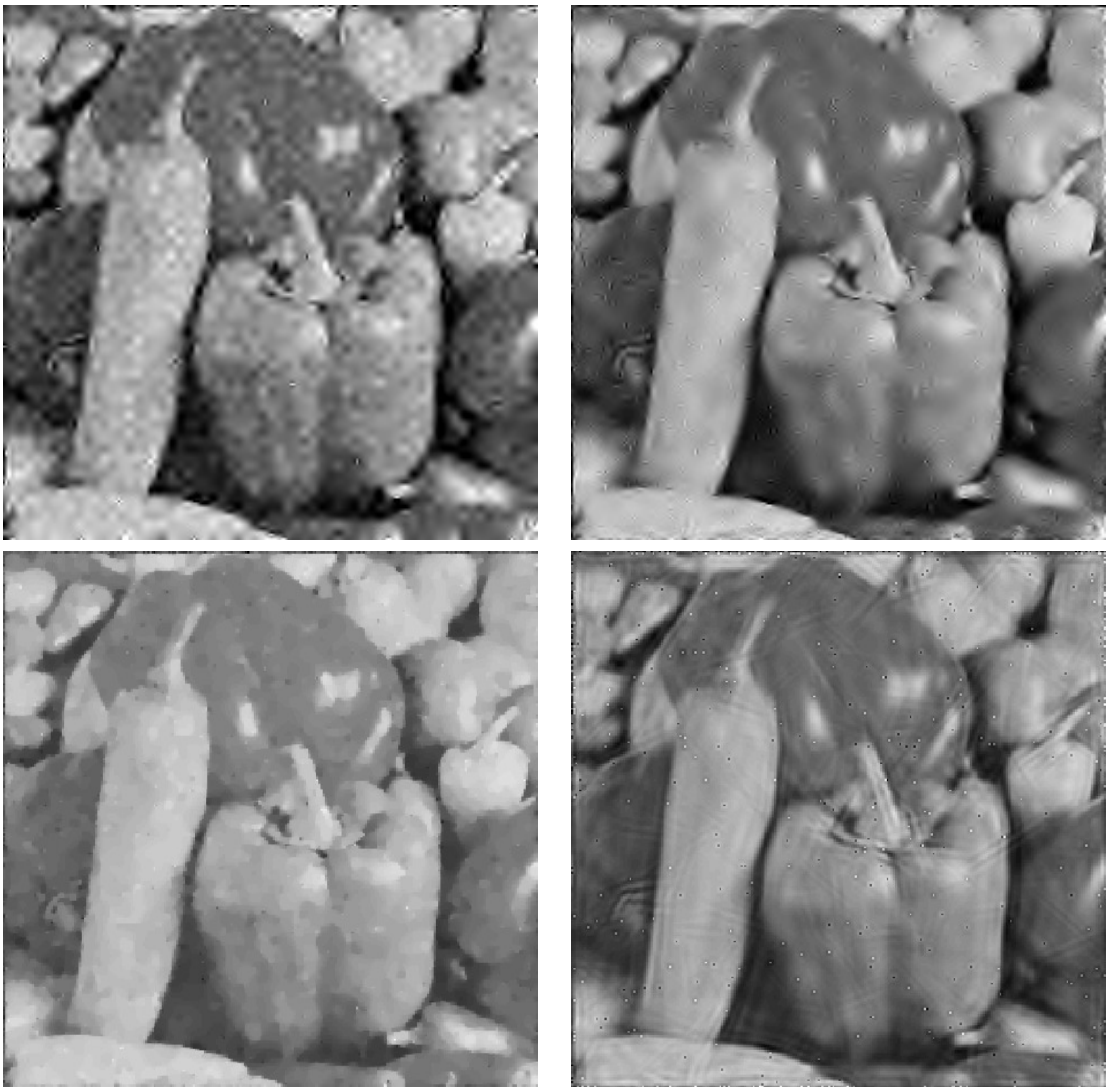

Abbildung 2.8: Entstörung des „Peppers“-Bildes ($\sigma = 0.15$) mittels Tensorprodukt-Wavelets ohne (oben links) und mit Cycle-Spinning (oben rechts), Vier-Pixel-Schema (unten links) und Curvelets (unten rechts).

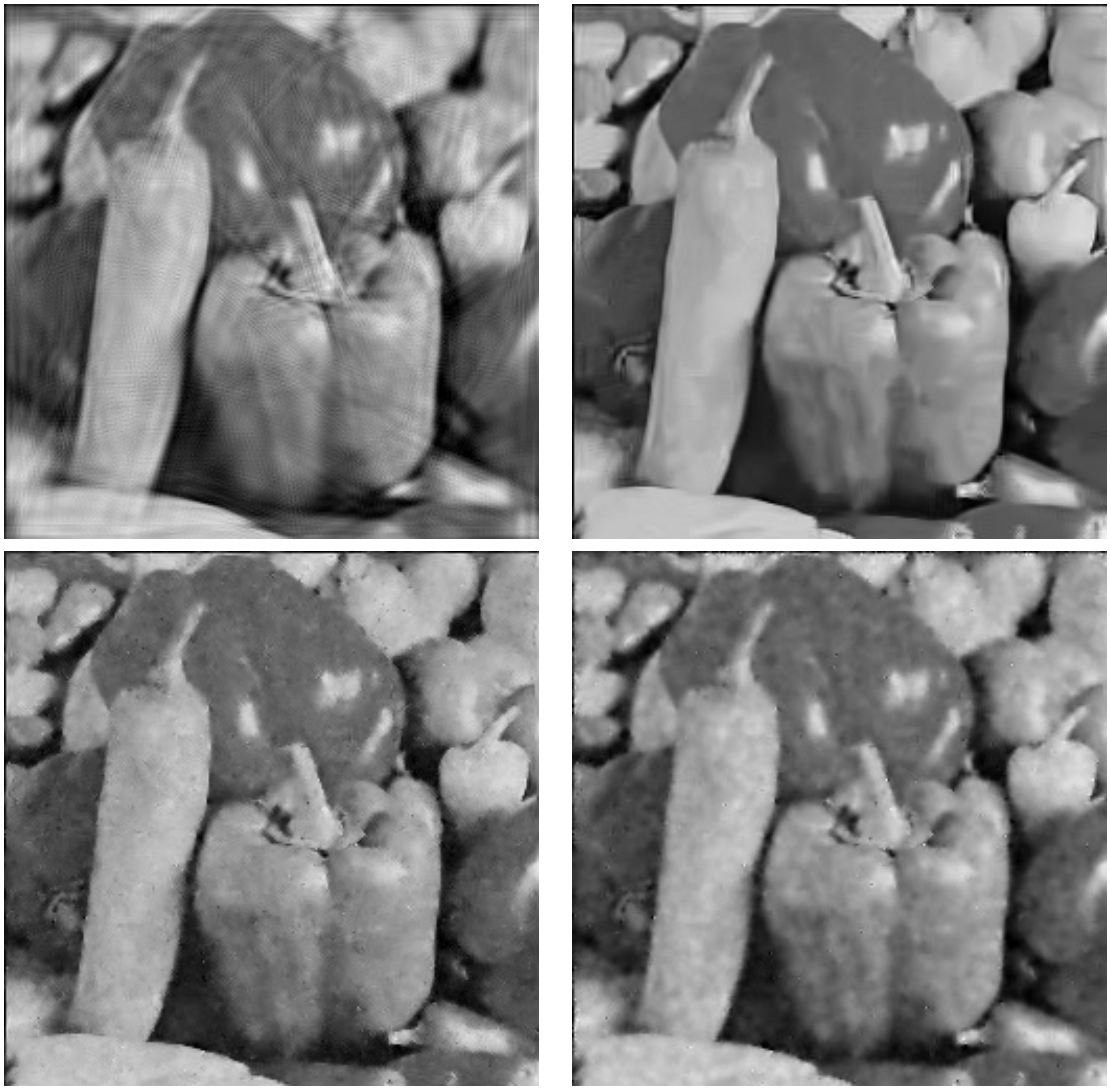

Abbildung 2.9: Entstörung des „Peppers“-Bildes ($\sigma = 0.15$) mittels Shearlets (oben links), BM3D (oben rechts), Algorithmus 2.63 mit adaptiv deterministischer (unten links) und adaptiv zufälliger Pfadkonstruktion (unten rechts).

Abbildung 2.10: Entstörung des „Cameraman“-Bildes ($\sigma = 0.1$) mittels Tensorprodukt-Wavelets ohne (oben links) und mit Cycle-Spinning (oben rechts), Vier-Pixel-Schema (unten links) und Curvelets (unten rechts).

Abbildung 2.11: Entstörung des „Cameraman“-Bildes ($\sigma = 0.1$) mittels Shearlets (oben links), BM3D (oben rechts), Algorithmus 2.63 mit adaptiv deterministischer (unten links) und adaptiv zufälliger Pfadkonstruktion (unten rechts).

Abbildung 2.12: Entstörung des „Cameraman“-Bildes ($\sigma = 0.15$) mittels Tensorprodukt-Wavelets ohne (oben links) und mit Cycle-Spinning (oben rechts), Vier-Pixel-Schema (unten links) und Curvelets (unten rechts).

Abbildung 2.13: Entstörung des „Cameraman“-Bildes ($\sigma = 0.15$) mittels Shearlets (oben links), BM3D (oben rechts), Algorithmus 2.63 mit adaptiv deterministischer (unten links) und adaptiv zufälliger Pfadkonstruktion (unten rechts).

	„Peppers“	„Peppers“	„Cameraman“	„Cameraman“
gestört	19.97	16.45	19.97	16.45
Tensorprodukt-Wavelet	24.91	23.20	24.74	22.86
Tensorprodukt-Wavelet mit Cycle-Spinning	28.11	25.85	27.19	25.14
Vier-Pixel-Schema	28.26	26.13	27.64	25.73
Curvelet	26.36	23.95	25.48	23.73
Shearlet	26.82	25.04	26.07	24.23
BM3D	30.22	28.18	29.36	27.49
Algorithmus 2.63, adaptiv deterministische Pfade	29.01	26.44	28.28	26.15
Algorithmus 2.63, adaptiv zufällige Pfade	27.96	25.69	27.44	24.85

Tabelle 2.6: Vergleich der PSNR-Werte bei der Entstörung mittels der jeweiligen Methoden.

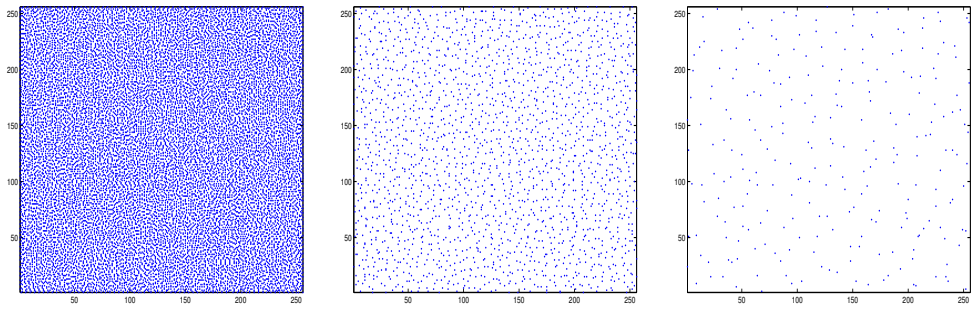

Abbildung 2.14: Aktualisierte gestreute Punktmengen Γ^2 (links), Γ^5 (Mitte) und Γ^8 (rechts) des „Cameraman“-Bildes bei adaptiv deterministischer Pfadkonstruktion.

Abbildung 2.15: L-förmiger Ausschnitt des „Cameraman“-Bildes: Mit Gaußschem Rauschen ($\sigma = 0.1$) belegte (links, PSNR = 19.96) und mittels Algorithmus 2.63 mit adaptiv deterministischer Pfadkonstruktion entrauschte Version (rechts, PSNR = 27.77).

auch bei den hier betrachteten Bildern (d.h. äquidistante Punkte im „nullten“ Level) ergeben sich in höheren Leveln nicht äquidistante Punktkonfigurationen.

Man beachte, dass sich der Einsatz der vorgeschlagenen Entstörungsmethode mittels Wavelets entlang von Pfaden nicht auf rechteckige Mengen beschränkt. Sie lässt sich zur Entstörung von auf nahezu beliebig geformten Mengen gegebenen Funktionswerten verwenden. Die Abbildungen 2.15-2.17 illustrieren die Entstörung eines L-förmigen und eines dreieckigen Ausschnitts des „Cameraman“-Bildes sowie die Entstörung des „Cameraman“-Bildes mit einem rautenförmig ausgestanzten Loch. Dabei wurde die adaptiv deterministische Pfadkonstruktion mit denselben Parametern wie in Tabelle 2.5 verwendet. Die angegebenen PSNR-Werte beziehen sich dabei auf die verringerte Anzahl an Pixeln.

Aufgrund der einfachen Visualisierung decken die hier vorgestellten numerischen Resultate nur den Fall zweidimensionaler Punktmengen ab. Grundsätzlich ist die Entstörungsmethode auch für Punktmengen in höheren Dimensionen nutzbar, wobei der numerische Aufwand mit der Dimension (wie auch mit der Anzahl der Datenpunkte) wächst.

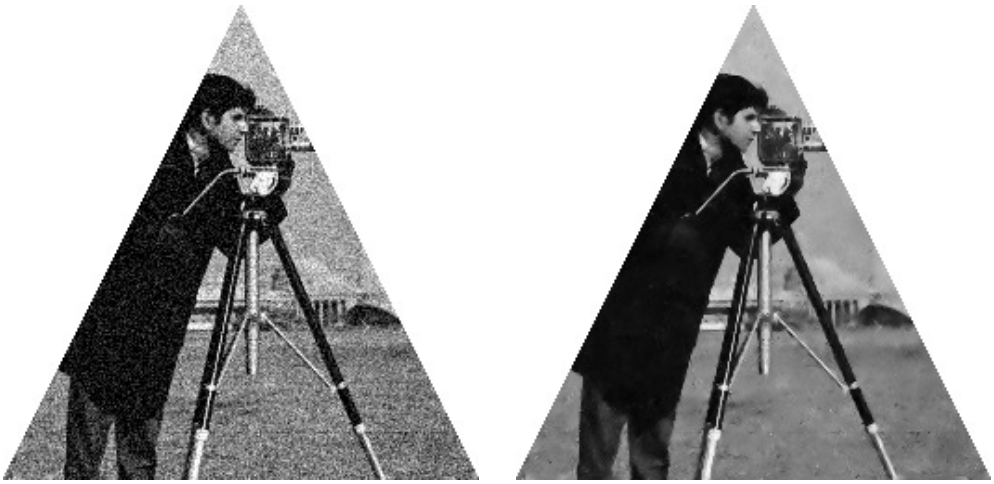

Abbildung 2.16: Dreieckiger Ausschnitt des „Cameraman“-Bildes: Mit Gaußschem Rauschen ($\sigma = 0.1$) belegte (links, PSNR = 19.98) und mittels Algorithmus 2.63 mit adaptiv deterministischer Pfadkonstruktion entrauschte Version (rechts, PSNR = 26.31).

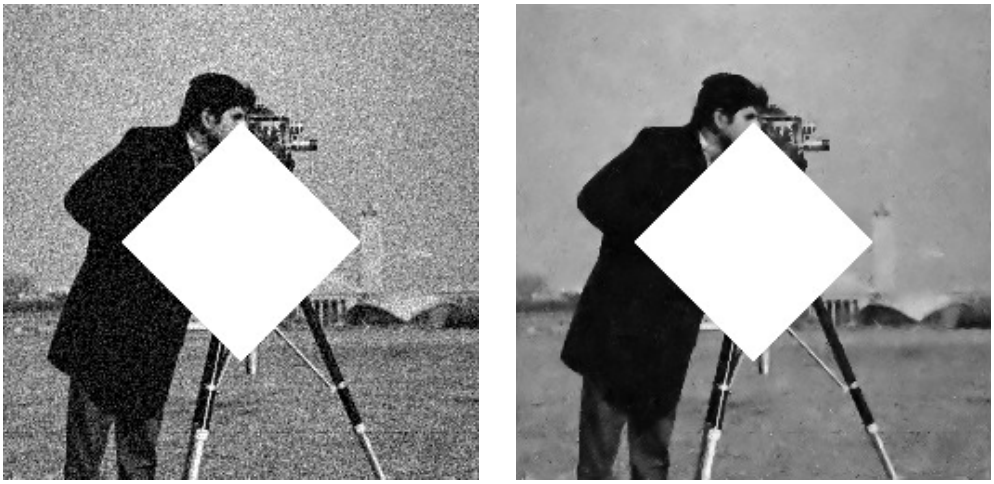

Abbildung 2.17: „Cameraman“-Bild mit Loch: Mit Gaußschem Rauschen ($\sigma = 0.1$) belegte (links, PSNR = 19.96) und mittels Algorithmus 2.63 mit adaptiv deterministischer Pfadkonstruktion entrauschte Version (rechts, PSNR = 28.71).

2.8 Modifikationen des Entstörungsalgorithmus 2.63

Wir widmen uns abschließend noch möglichen Modifikationen des Entstörungsalgorithmus 2.63. In den Unterabschnitten 2.8.1 und 2.8.2 werden wir Vereinfachungen vorstellen, welche den numerischen Aufwand reduzieren, jedoch nicht die Entstörungsqualität des ursprünglichen Algorithmus 2.63 erreichen. Der Ansatz in Unterabschnitt 2.8.3 sorgt für eine höhere Effizienz der Methode bei Erhaltung der Entstörungsgüte. Ziel der in den Unterabschnitten 2.8.4 und 2.8.5 diskutierten Modifikationen war eine weitere Verbesserung der Entstörungsergebnisse. Auch die in Unterabschnitt 2.8.6 beschriebene Modifikation zielte vorrangig auf eine Erhöhung der Entstörungsgüte ab.

2.8.1 Einmalige Pfadkonstruktion

Der Entstörungsalgorithmus 2.63 zeichnet sich u.a. dadurch aus, dass in jedem Zerlegungslevel ein neuer Pfad (auf dem aktualisierten Datensatz) konstruiert wird. Der numerische Aufwand lässt sich daher wesentlich reduzieren, wenn wir nur einmalig im ersten Level einen Pfad durch die gegebenen gestreuten Punkte $x_1, \ldots, x_N$ konstruieren. Für die weiteren Level verwenden wir die durch den Pfad des ersten Level induzierte Umordnung. Eine entsprechende Vereinfachung der EPWT für die Bildkompression wird in [130] von Tenorth unter dem Namen „Einfach-EPWT“ erwähnt.

Algorithmus 2.75 Entstörung mit einmaliger Pfadkonstruktion
Gegeben: *$x_j \in \mathbb{R}^d$, $\tilde{f}(x_j)$ für $j = 1, \ldots, N$, wobei $N = 2^t$ für ein $t \in \mathbb{N}$, und eine biorthogonale Wavelet-Filterbank mit Zerlegungsfiltern h, g und Rekonstruktionsfiltern $\tilde{h}, \tilde{g}$.*

Führe die folgenden zwei Schritte durch:

1. *Konstruiere einen geeigneten Pfadvektor $p \in \mathbb{R}^N$, d.h. eine Permutation der Indizes $\{1, \ldots, N\}$, welche eine Reihenfolge der Punkte $x_{p(j)}$ und zugeordneten Funktionswerte $\tilde{f}(x_{p(j)})$ darstellt.*

2. *Wende die aus aus den Filtern $h, g, \tilde{h}, \tilde{g}$ bestehende iterative Wavelet-Filterbank mit Shrinkage und $L < t$ Transformationsleveln auf das Eingangssignal $(\tilde{f}(x_{p(j)})_{j=1}^N$ an.*

Ausgabe: *Den gestreuten Punkten x_j zugeordnete geglättete Funktionswerte.*
Iteration: *Wiederhole die Schritte 1 und 2 (unter Verwendung eines neuen Pfadvektors) T-mal (z.B. $T = 64$) und bilde den Durchschnitt der Ausgaben.*

Bei diesem modifizierten Vorgehen sind sowohl der Multiskalencharakter als auch die Adaptivität der ursprünglichen Methode nur abgeschwächt vorhanden. Durch die Beibehaltung des Pfades für jedes Level wird die EPWT auf eine echt eindimensionale Transformation (entlang dieses Pfades) reduziert und kann daher Datenkorrelationen nur in Richtung dieses Pfades ausnutzen. Erwartungsgemäß sinkt der numerische Aufwand, aber auch die Qualität der Entstörung wie sich exemplarisch an der Entstörung des „Cameraman"-Bildes (siehe Abbildung 2.18 und Tabelle 2.8) ablesen lässt. Für dieses Beispiel wurde die adaptiv deterministische Pfadkonstruktion mit 6 Zerlegungsleveln, Threshold-Parameter $\theta = 0.35$ sowie Nachbarschaftsparameter $C_1 = \sqrt{2}$ (siehe (2.2)) gewählt. Die Laufzeit des Algorithmus kann gegenüber der standardmäßigen Verwendung von Algorithmus 2.63 auf weniger als ein Sechstel verringert werden.

2.8.2 Nicht adaptive Pfadkonstruktion

Weiterhin denkbar ist eine Verwendung des ursprünglichen Algorithmus 2.63 unter Verwendung einer nicht adaptiven Pfadkonstruktion anstelle der adaptiven Konstruktionen gemäß der Algorithmen 2.67 oder 2.69. Dabei berücksichtigen wir nur die spatialen Abstände der Datenpunkte x_j^l, $j = 1, \ldots, N/2^l$, im jeweiligen Zerlegungslevel, jedoch nicht die zugehörigen Funktions- bzw. Tiefpass-Werte c_j^l.

Eine nicht adaptive Version der deterministischen Pfadkonstruktion nach Algorithmus 2.67 im l-ten Level sieht man im folgenden Algorithmus.

Algorithmus 2.76 Nicht adaptiv deterministische Pfadkonstruktion
Gegeben: $\Gamma^l = \{x_1^l, \ldots, x_{N/2^l}^l\} \subset \mathbb{R}^d$.

1. *Wähle den ersten Index des Pfades $p^l(1)$ zufällig aus $\{1, \ldots, N/2^l\}$ unter der Annahme, dass alle Indizes die gleiche Wahrscheinlichkeit besitzen, gewählt zu werden.*

2. *Für $k = 1, \ldots, N/2^l - 1$ durchlaufe folgende Schritte:*

   a) *Berechne $N_{C_1}(x_j^l)$ und $\tilde{N}_{C_1}(x_{p^l(k)}^l)$ gemäß (2.1) und (2.2).*

   b) *Wähle den nächsten Index $p^l(k+1)$ des Pfades wie folgt:*

   - *Falls $\tilde{N}_{C_1}(x_{p^l(k)}^l)$ nicht die leere Menge ist, wähle $p^l(k+1)$, sodass*

$$x_{p^l(k+1)}^l = \underset{x \in \tilde{N}_{C_1}(x_{p^l(k)}^l)}{\operatorname{argmax}} \frac{\langle x_{p^l(k-1)}^l - x_{p^l(k)}^l, x_{p^l(k)}^l - x \rangle}{\|x_{p^l(k-1)}^l - x_{p^l(k)}^l\|_2 \cdot \|x_{p^l(k)}^l - x\|_2} \tag{2.11}$$

für $k > 1$. Für $k = 1$ wähle $p^l(k+1)$ zufällig unter den Indizes der Punkte in $N_{C_1}(x^l_{p^l(k)})$ unter der Annahme einer Gleichverteilung.

- *Falls $\tilde{N}_{C_1}(x^l_{p^l(k)})$ leer ist, wähle $p^l(k+1)$ zufällig unter den Indizes der Punkte in $\tilde{N}_{k,l}$ (siehe (2.4)) unter der Annahme einer Gleichverteilung.*

3. **Ausgabe:** *Pfadvektor* $p^l = (p^l(k))_{k=1}^{N/2^l}$.

Bemerkung 2.77 *Die Bedingung* (2.11) *sichert erneut, dass der Pfad eine eingeschlagene Richtung beibehält, solange dies möglich ist. Der Algorithmus 2.76 führt somit in der Regel zu „schneckenförmigen“ Pfaden.*

Für eine nicht adaptive Abwandlung der zufälligen Pfadkonstruktion können wir Algorithmus 2.69 übernehmen und müssen lediglich die Definition der Übergangswahrscheinlichkeiten abändern, indem wir nur die spatialen Informationen berücksichtigen. Statt der Gewichte $w(y_i^l, y_j^l)$ wie in (2.8), welche durch entsprechende Normalisierung die Übergangswahrscheinlichkeiten darstellen, verwenden wir dann rein spatiale Gewichte

$$w(x_i^l, x_j^l) = w^1(x_i^l, x_j^l) = \exp\left(\frac{-\|x_i^l - x_j^l\|_2^2}{2^{2l/d}\eta_1}\right).$$

Der Verzicht auf Adaptivität der Pfadkonstruktion verschlechtert die Entstörungsergebnisse (siehe Abbildung 2.18 und Tabelle 2.8), beschleunigt aber die Prozedur, da eine Vorberechnung der Pfade möglich wird . Die eigentliche Entstörungsprozedur läuft dann ähnlich schnell wie eine übliche Wavelet-Transformation. Für das erwähnte numerische Beispiel wurden der Threshold-Parameter $\theta = 0.35$ und der Nachbarschaftsparameter $C_1 = 2$ (siehe (2.2)) sowie 4 Zerlegungslevel verwendet.

Es sei jedoch angemerkt, dass die Laufzeit der nicht adaptiv deterministischen Pfadkonstruktion (Algorithmus 2.76) selbst verglichen mit der adaptiv deterministischen Pfadkonstruktion (Algorithmus 2.67) nur minimal geringer ist. Die Verringerung der Laufzeit der Entstörungsprozedur ergibt sich nur, wenn man die Möglichkeit der Vorberechnung der Pfade ausnutzt. Wir erwähnen hier eine weitere nicht adaptive Pfadkonstruktion, welche noch weniger komplex ist.

Algorithmus 2.78 Primitive Pfadkonstruktion
Gegeben: $\Gamma^l = \{x_1^l, \ldots, x_{N/2^l}^l\} \subset \mathbb{R}^d$.

1. *Wähle den ersten Index des Pfades $p^l(1)$ zufällig aus $\{1, \ldots, N/2^l\}$ unter der Annahme, dass alle Indizes die gleiche Wahrscheinlichkeit besitzen, gewählt zu werden.*

2. *Für* $k = 1, \ldots, N/2^l - 1$ *durchlaufe folgende Schritte:*

   a) *Berechne* $N_{C_1}(x_j^l)$ *und* $\tilde{N}_{C_1}(x^l_{p^l(k)})$ *gemäß* (2.1) *und* (2.2).

   b) *Wähle den nächsten Index* $p^l(k+1)$ *des Pfades wie folgt:*

   - *Falls* $\tilde{N}_{C_1}(x^l_{p^l(k)})$ *nicht die leere Menge ist, wähle* $p^l(k+1)$ *zufällig unter den Indizes der Punkte in* $N_{C_1}(x^l_{p^l(k)})$ *unter der Annahme einer Gleichverteilung.*

   - *Falls* $\tilde{N}_{C_1}(x^l_{p^l(k)})$ *leer ist, wähle* $p^l(k+1)$ *zufällig unter den Indizes der Punkte in* $\tilde{N}_{k,l}$ *(siehe* (2.4)*) unter der Annahme einer Gleichverteilung.*

3. **Ausgabe:** *Pfadvektor* $p^l = (p^l(k))_{k=1}^{N/2^l}$.

Die primitive Pfadkonstruktion gleicht der nicht adaptiv deterministischen Pfadkonstruktion, verzichtet aber auf die Bedingung (2.11), welche fördert, dass aufeinanderfolgende Pfadpunkte möglichst auf einer Geraden liegen. Stattdessen wird der nächste Pfadpunkt zufällig gemäß einer Gleichverteilung unter den spatialen Nachbarn des aktuellen Punktes festgelegt. In der Praxis reduziert sich die Laufzeit gegenüber der adaptiv deterministischen Pfadkonstruktion auf fast die Hälfte, wohingegen die Güte der Entstörung noch etwas geringer als bei der Verwendung der nicht adaptiv deterministischen Pfadkonstruktion ist.

2.8.3 Zerlegung des Datensatzes

Die für die Pfadkonstruktion gemäß Algorithmus 2.67 oder 2.69 benötigte Rechenzeit wächst stark mit der Anzahl N der Punkte des vorliegenden Datensatzes. Für größere Datensätze wird der Entstörungsalgorithmus 2.63 somit sehr zeitintensiv. Als Gegenmittel lässt sich die Menge der Abtastpunkte $\Gamma \subset \mathbb{R}^d$ zerlegen und die Entstörung gemäß Algorithmus 2.63 nacheinander auf den Teilmengen der Zerlegung durchführen. Hierdurch dämpft man das Wachstum der Rechenzeit bei steigender Anzahl N der Punkte ab.

Es sei wieder $\{x_1, \ldots, x_N\} = \Gamma \subset \Omega$, wobei Ω sich als Vereinigung endlich vieler beschränkter, zusammenhängender Teilmengen des $\mathbb{R}^d$ schreiben lässt. Wir wählen nun eine Zerlegung von Ω in paarweise disjunkte Teilmengen Ω_s, $s = 1, \ldots, S$, mit

$$\cup_{s=1}^{S} \Omega_s = \Omega$$

für ein $S \in \mathbb{N}$. Wir definieren dann

$$\Gamma_s := \{x \in \Gamma : x \in \Omega_s\}$$

und wenden den Entstörungsalgorithmus 2.63 nacheinander auf $\Gamma_1, \ldots, \Gamma_S$ an.

Bemerkung 2.79 *1. Derartige Ansätze, bei denen man die betrachtete Definitionsmenge in kleinere Teilmengen zerlegt und den gewählten Algorithmus auf diesen Teilmengen anwendet, sind in unterschiedlichsten Richtungen der Mathematik zur Beschleunigung von Algorithmen verbreitet. Exemplarisch sei hier erwähnt, dass Dekel und Leviatan [38] bzw. Dekel und Nemirosky [39] bei der Anwendung der von ihnen propagierten geometrischen Wavelets einen solchen Zerlegungsansatz vorschlagen.*

2. *Es ist zweckmäßig, die Zerlegung so zu bilden, dass die Teilmengen Γ_s alle ähnliche Kardinalität besitzen. Für die Anwendung auf ein Bild mit 256×256 Pixeln ist etwa eine Zerlegung in 16 Blöcke mit je 64×64 Pixeln denkbar.*

3. *Die Teilmengen Γ_s der Zerlegung dürfen natürlich nicht zu klein gewählt werden, da sich die Strukturen innerhalb des Datensatzes, denen die konstruierten Pfade folgen, sonst nicht mehr ausnutzen lassen, was zu einer Verschlechterung der Entstörungsergebnisse führt. Weiterhin steigt die Rechenzeit des Gesamtalgorithmus bei zu kleinen (und damit zu zahlreichen) Teilmengen wieder an.*

4. *Eine weitere Verkürzung der Rechenzeit kann erreicht werden, wenn man die Entstörung der Teilmengen parallel statt nacheinander laufen lässt.*

5. *Für die mögliche Übertragung auf die EPWT für Kompressionszwecke sei bemerkt, dass der Ansatz mit der Zerlegung des Datensatzes zu einer Erhöhung des Speicheraufwandes für die Pfade führen kann.*

In numerischen Tests verringert sich die Rechenzeit gegenüber dem Vorgehen ohne Zerlegung auf fast ein Drittel und gleichzeitig wird ein etwa gleichwertiger PSNR-Wert erreicht (siehe Abbildung 2.18 und Tabelle 2.8). Für dieses Resultat haben wir die adaptiv deterministische Pfadkonstruktion genutzt und die freien Parameter wurden wie in Tabelle 2.5 gesetzt.

Bemerkung 2.80 *Der beschriebene Zerlegungsansatz besticht durch seine Einfachheit. Man kann komplexere Weiterentwicklungen in folgende Richtungen in Betracht ziehen:*

1. *Es lassen sich unterschiedliche Zerlegungen für die unterschiedlichen Iterationen des Algorithmus 2.63 wählen. Für ein Bild sind beispielsweise neben der angesprochenen Zerlegung in Blöcke Zerlegungen in vertikale und horizontale „Streifen“*

denkbar (vgl. [39]). Dadurch, dass sich Teilmengen der unterschiedlichen Zerlegungen überlappen, wird der Ansatz bei der Durchschnittsbildung robuster und mögliche Artefakte an den Rändern der Teilmengen werden vermindert.

2. *Der Zerlegungsansatz kann levelweise befolgt werden. Man beachte, dass sich die Kardinalität der aktualisierten Punktmengen Γ^l in Algorithmus 2.63 von Level zu Level halbiert. Wenn man benachbarte Teilmengen der Zerlegung im nächsthöheren Level vereinigt, so bleibt die durchschnittliche Kardinalität der Teilmengen folglich von Level zu Level gleich. Dieses Vorgehen wirkt ebenfalls Artefakten an den Rändern der Teilmengen entgegen.*

3. *Wir betrachten bisher nur nicht adaptive Zerlegungen, welche die den Punkten zugeordneten Funktionswerte nicht berücksichtigen. Optimal wären adaptive Zerlegungen, die sich nach den Glattheitsstrukturen im Datensatz richten. Der Aufwand zur Bestimmung passender derartiger Zerlegungen ist jedoch hoch.*

2.8.4 Pfade mit redundanten Punkten

Einen möglichen Nachteil beim vorgeschlagenen Entstörungsalgorithmus 2.63 stellen die „Sprünge“ bei der Pfadkonstruktion (siehe Bemerkung 2.68 bzw. 2.70) dar, da hier die Bedingung, dass aufeinanderfolgende Pfadpunkte spatial benachbart sein sollen, fallengelassen wird. Ein denkbarer Ansatz, das Auftreten dieser „Sprünge“ bei der Pfadkonstruktion zu minimieren, besteht darin, den nächsten Punkt stets unter den spatialen Nachbarn des aktuellen Punktes zu wählen, auch wenn bereits alle Nachbarn vom Pfad besucht wurden. Man lässt ein mehrfaches Vorkommen von Punkten auf dem Pfad zu.

Wir beschreiben zunächst einen Zerlegungsschritt von Level l zu Level $l+1$. Wie erwähnt verzichten wir auf die Bedingung, dass jeder Punkt x_j^l genau einmal durch den Pfad besucht wird. Noch nicht besuchte Punkte der Nachbarschaft des aktuellen Pfadpunktes sollen bevorzugt als nächster Punkt des Pfades gewählt werden, doch mehrfaches Auftreten eines Punktes ist erlaubt. Andererseits kann es somit vorkommen, dass Punkte überhaupt nicht besucht werden.

Wir konstruieren einen Pfad der Länge $2N$. Hierdurch wird zusätzliche Redundanz eingeführt, welche im Hinblick auf Entstörung hilfreich sein kann, und die Wahrscheinlichkeit reduziert, dass Punkte gar nicht im Pfad auftreten. Jeder Punkt x_j^l verfügt demnach nach der Pfadkonstruktion über eine Vielfachheit $\nu_j^l \in \{0, 1, \ldots, 2N\}$.

Wir wenden die Zerlegungsfilter einer gegebenen Wavelet-Filterbank mit anschließendem Downsampling auf die Tiefpass-Werte c_j^l entlang des Pfades, d.h. auf den Vektor $(c^l_{p^l(j)})_{j=1}^{2N}$, an. Weiterhin wenden wir einen Tiefpass-Filter mit anschließendem Down-

sampling komponentenweise auf die Punkte x_j^l entlang des Pfades, d.h. auf $(x^l_{p^l(j)})_{j=1}^{2N}$, an. Wir ordnen die resultierenden Tiefpass-Werte $(c_j^{l+1})_{j=1}^N$ den resultierenden Punkten $(x_j^{l+1})_{j=1}^N$ zu und erhalten so einen aktualisierten Datensatz. Man beachte, dass wir im Hintergrund neben den Hochpass-Koeffizienten diesmal auch die ursprünglichen Tiefpass-Koeffizienten c_j^l, die den Indizes zugeordnet werden, welche nicht Teil des Pfades p^l sind, speichern müssen.

Bei der Rekonstruktion von Level l aus Level $l+1$ ergeben sich folglich ν_j^l Rekonstruktionen für den dem Punkt x_j^l zugeordneten Tiefpass-Wert. Für $\nu_j^l \geq 1$ verwenden wir das arithmetische Mittel dieser Rekonstruktionswerte. Für $\nu_j^l = 0$ greifen wir auf die unbehandelten Tiefpass-Werte zurück, welche wir zuvor im Hintergrund gespeichert hatten. Algorithmus 2.81 fasst die Idee zusammen.

Algorithmus 2.81 Entstörung mit redundanten Pfadpunkten
Gegeben: $\Gamma = \{x_1, \ldots, x_N\} =: \{x_1^0, \ldots, x_N^0\} =: \Gamma^0 \subset \mathbb{R}^d$, $\tilde{f}(x_j) =: c_j^0$ *für* $j = 1, \ldots, N$, *wobei* $N = 2^t$ *für ein* $t \in \mathbb{N}$, *und eine biorthogonale Wavelet-Filterbank mit Zerlegungsfiltern* h, g *und Rekonstruktionsfiltern* $\tilde{h}, \tilde{g}$ *mit* $\sum_{k\in\mathbb{Z}} h_k = \sqrt{2}$ *sowie ein Tiefpass-Filter* γ *mit* $\sum_{k\in\mathbb{Z}} \gamma_k = 1$.

Zerlegung: *Führe die folgenden vier Schritte für* $l = 0, 1, \ldots, L-1$ *mit* $L < t$ *durch:*

1. *Konstruiere einen geeigneten Pfadvektor* $p^l = (p^l(j))_{j=1}^{2N}$ *mit redundanten Indizes* $p^l(j)$ *mit den Vielfachheiten* ν_j^l. *Speichere die Tiefpass-Koeffizienten* c_j^l *für Indizes* j *mit* $\nu_j^l = 0$.

2. *Wende den (periodischen) Tiefpass-Filter* $(\bar{h}_{-k})_{k\in\mathbb{Z}}$ *bzw. den (periodischen) Hochpass-Filter* $(\bar{g}_{-k})_{k\in\mathbb{Z}}$ *auf den Vektor* $(c^l_{p^l(j)})_{j=1}^{2N}$ *an und erhalte nach Downsampling die Tiefpass-Koeffizienten* $(c_j^{l+1})_{j=1}^N$ *bzw. die Hochpass-Koeffizienten* $(d_j^{l+1})_{j=1}^N$.

3. *Wende den Tiefpass-Filter* $(\bar{\gamma}_{-k})_{k\in\mathbb{Z}}$ *auf den Vektor* $(x^l_{p^l(j)})_{j=1}^{2N}$ *(separat für jede der* d *Komponenten) an und erhalte nach Downsampling die aktualisierten gestreuten Punkte* $(x_j^{l+1})_{j=1}^N$. *Setze* $\Gamma^{l+1} := \{x_1^{l+1}, \ldots, x_N^{l+1}\}$.

4. *Führe ein Hard-Shrinkage mit einem Parameter* $\theta > 0$ *an den Hochpass-Koeffizienten* $(d_j^{l+1})_{j=1}^N$ *resultierend in*

$$\tilde{d}_j^{l+1} = \begin{cases} d_j^{l+1} & \text{falls } |d_j^{l+1}| \geq \theta \\ 0 & \text{falls } |d_j^{l+1}| < \theta \end{cases}$$

   durch.

Rekonstruktion: *Initialisiere* $(\hat{c}_j^L)_{j=1}^N := (c_j^L)_{j=1}^N$ *und führe die folgenden vier Schritte für* $l = L, L-1, \ldots, 1$ *durch:*

5. *Wende Upsampling und anschließend den Tiefpass-Filter $\tilde{h}$ auf $(\tilde{c}_j^l)_{j=1}^N$ an.*

6. *Wende Upsampling und anschließend den Hochpass-Filter $\tilde{g}$ auf $(\tilde{d}_j^l)_{j=1}^N$ an.*

7. *Addiere die Ergebnisse von Schritt 5 und 6, um den Vektor $(\tilde{c}^{l-1}_{p^{l-1}(i),\mu_i^{l-1}})_{i=1}^{2N}$ zu erhalten. Dieser Vektor enthält die jeweils ν_j^{l-1} Rekonstruktionswerte für jeden Tiefpass-Wert c_j^{l-1}. Dabei gibt*

$$\mu_i^{l-1} := |\{r = 1, \dots, i : p^{l-1}(r) = p^{l-1}(i)\}|$$

an, wie oft der Index $p^{l-1}(i)$ vom ersten bis zum i-ten Bestandteil des Pfades vorkommt (siehe auch Punkt 5 in Bemerkung 2.82).

8. *Berücksichtige die Vielfachheiten ν_j^{l-1} und setze*

$$\hat{c}_j^{l-1} = \begin{cases} \frac{1}{\nu_j^{l-1}} \sum_{\nu=1}^{\nu_j^{l-1}} \tilde{c}_{j,\nu}^{l-1} & \text{für } \nu_j^{l-1} > 0 \\ c_j^{l-1} & \text{für } \nu_j^{l-1} = 0 \end{cases}.$$

Ausgabe: *Den gestreuten Punkten $x_j \in \Gamma$ zugeordnete geglättete Funktionswerte $(\hat{c}_j^0)_{j=1}^N$.*
Iteration: *Wiederhole die Schritte 1 bis 8 (unter Verwendung neuer Pfadvektoren) T-mal (z.B. $T = 64$) und bilde den Durchschnitt der Ausgaben $(\hat{c}_j^0)_{j=1}^N$.*

Bemerkung 2.82 1. *Für $l \geq 1$ ist Γ^l als Multimenge zu verstehen. Es können also mehrere Punkte mit identischen spatialen Koordinaten, d.h. $x_i^l = x_j^l$ für $i \neq j$, existieren. Insbesondere wird dieser Fall offensichtlich regelmäßig auftreten, wenn man die Aktualisierung der Punkte in Schritt 3 auf ein reines Downsampling beschränkt.*

2. *Man beachte, dass im Gegensatz zur ursprünglichen Idee in Algorithmus 2.63 mit Voranschreiten der Zerlegungslevel keine Ausdünnung der Punktmenge Γ^l erfolgt. Es gilt $|\Gamma^l|$=N für alle $l = 0, \dots, L-1$. Hierdurch geht ein Teil der arithmetischen Effizienz der Wavelet-Transformation verloren. Weiterhin bleibt der Aufwand für die Pfadkonstruktion von Level zu Level gleich, wodurch sich der Aufwand für den gesamten Algorithmus stark erhöht. Aus diesem Grund kann es sinnvoll sein, für Algorithmus 2.81 weniger komplexe (z.B. nicht adaptive) Pfadkonstruktionen (siehe Algorithmus 2.85) oder eine Zerlegung des Datensatzes (siehe Unterabschnitt 2.8.3) in Betracht zu ziehen.*

3. *Eine Ausdünnung des Datensatzes lässt sich erreichen, indem man im nächsten Level für mehrfach vorkommende Indizes das arithmetische Mittel der entsprechenden Tiefpass-Werte ansetzt. Es ist zu beachten, dass wir hierbei eine zusätzliche*

Fehlerquelle bei der Rekonstruktion erhalten, da das Mitteln nicht invertierbar ist. Weiterhin ist nicht mehr gewährleistet, dass $|\Gamma^l|$ eine Zweier-Potenz bleibt. Aufgrund dieser Komplikationen verfolgen wir diesen Ansatz nicht weiter.

4. *Grundsätzlich erlaubt die Idee, Pfade mit einer Länge ρN (mit $\rho \in \mathbb{N}_{>2}$) zu betrachten. Durch die höhere Redundanz steigt das Entstörungspotenzial der Methode. Die angesprochenen Nachteile kommen jedoch noch stärker zum Tragen, da sich die Kardinalität von Γ^l mit wachsendem Level l sogar vergrößert. In numerischen Experimenten ergab die Wahl $\rho = 3$ oder $\rho = 4$ bei Betrachtung des Durchschnittes über $T = 64$ Iterationen gegenüber der Wahl $\rho = 2$ zudem keine besseren Entstörungsergebnisse.*

5. *Der in Schritt 7 erhaltene Vektor $(\tilde{c}^{l-1}_{p^{l-1}(i),\mu_i^{l-1}})_{i=1}^{2N}$ ist, wenn beispielsweise der Index $j \in \{1,\dots,N\}$ dreimal im Pfad p^{l-1} vorkommt, von der Form*

$$(\dots \tilde{c}^{l-1}_{j,1} \dots \tilde{c}^{l-1}_{j,2} \dots \tilde{c}^{l-1}_{j,3} \dots).$$

Eine mögliche Konstruktion eines adaptiv zufälligen Pfadvektors mit redundanten Punkten für Schritt 1 in Algorithmus 2.81 ist mit Algorithmus 2.83 gegeben.

Algorithmus 2.83 Adaptiv zufällige Pfadkonstruktion mit redundanten Punkten
Gegeben: $y_j^l = ((x_j^l)^T, c_j^l)^T \in \mathbb{R}^{d+1}$ *für* $j = 1,\dots,N$.

1. *Berechne die Gewichtsmatrix $W^l = (w(y_i^l, y_j^l))_{i,j=1}^N$ nach (2.8), wobei Gewichte für nicht spatial benachbarte Paare gemäß (2.9) zu Null gesetzt werden.*

2. *Wähle den ersten Index des Pfades $p^l(1)$ zufällig aus $\{1,\dots,N\}$ gemäß einer Gleichverteilung.*

3. *Durchlaufe folgende Iteration für $k = 1,\dots,2N-1$:*
   Nach der Konstruktion von $p^l(1),\dots,p^l(k)$ im Pfadvektor berechnen wir

$$s_{p^l(k)} = \sum_{\substack{j=1 \\ j \notin \bigcup_{j=1}^k \{p^l(j)\}}}^{N} w(y^l_{p^l(k)}, y^l_j)$$

   und bestimmen die Übergangswahrscheinlichkeiten $P_{p^l(k),r}$ vom aktuellen Index $p^l(k)$ zu Index r vermöge

$$P_{p^l(k),r} := \begin{cases} \frac{w(y^l_{p^l(k)}, y^l_r)}{s_{p^l(k)}} & \text{falls } r \in \{1,\dots,N\} \setminus \bigcup_{j=1}^k \{p^l(j)\} \text{ und } s_{p^l(k)} \neq 0 \\ 0 & \text{sonst} \end{cases}. \tag{2.12}$$

a) Ist $P_{p^l(k),r} \neq 0$ für mindestens ein $r \in \{1,\dots,N\}$, so wähle $p^l(k+1)$ zufällig gemäß der eingeführten Wahrscheinlichkeitsverteilung.

b) Ist $P_{p^l(k),r} = 0$ für alle $r \in \{1,\dots,N\}$, so wähle $p^l(k+1)$ zufällig gemäß einer Gleichverteilung unter den spatialen Nachbarn des aktuellen Index $p^l(k)$, d.h. unter den Indizes $r \in \{1,\dots,N\}$ mit $\|x^l_{p^l(k)} - x^l_r\|_2 \leq 2^{l/d} D_1$ für einen Parameter $D_1 > 0$.

Ausgabe: *Pfadvektor $p^l = (p^l(k))_{k=1}^{2N}$.*

Bemerkung 2.84 *1. Aufgrund der Bestimmung der Gewichte gemäß (2.9) im ersten Schritt können nur spatial benachbarte Punkte eine positive Übergangswahrscheinlichkeit erhalten. Durch die vorläufige Zuweisung der Übergangswahrscheinlichkeit Null für bereits verwendete Punkte in (2.12) werden noch nicht verwendete Nachbarpunkte gemäß dem Vorgehen in Schritt 3a) in jedem Fall vor der erneuten Wahl eines bereits verwendeten Nachbarpunktes gewählt. Andererseits sorgt das Vorgehen gemäß Schritt 3b) dafür, dass, wenn alle Nachbarpunkte bereits einmal gewählt wurden, einer der Nachbarpunkte ein zweites Mal gewählt wird und wir nicht zu einem Punkt außerhalb der Nachbarschaft springen.*

2. Der Nachbarschaftsparameter $D_1 > 0$ sollte so gewählt werden, dass jeder Punkt mindestens einen Nachbarpunkt besitzt. Ist dies nicht erfüllt, so wählen wir den nächsten Punkt des Pfades zufällig gemäß einer Gleichverteilung unter allen noch nicht verwendeten Punkten (sofern existent) bzw. unter allen Punkten (sofern jeder Punkt bereits verwendet wurde). In diesem Fall wird die spatiale Entfernung der Punkte nicht berücksichtigt. Außerhalb dieses Sonderfalls, d.h. bei geeigneter Festsetzung des Parameters D_1, ist der nächste Pfadpunkt $x^l_{p^l(k+1)}$ jedoch stets ein Nachbarpunkt des aktuellen Pfadpunktes $x^l_{p^l(k)}$.

3. Statt die Übergangswahrscheinlichkeit für bereits verwendete Punkte in (2.12) Null zu setzen, lassen sich die Übergangswahrscheinlichkeiten auch vermöge

$$P_{p^l(k),r} := \begin{cases} \frac{\tilde{w}(y_{p^l(k)}, y^l_r)}{\tilde{s}_{p^l(k)}} & \text{falls } \tilde{s}_{p^l(k)} \neq 0 \\ 0 & \text{sonst} \end{cases}$$

mit

$$\tilde{w}(y^l_{p^l(k)}, y^l_r) := \frac{w(y^l_{p^l(k)}, y^l_r)}{B^{\mu^l(k,r)}}$$

und

$$\tilde{s}_{p^l(k)} := \sum_{j=1}^{N} \tilde{w}(y^l_{p^l(k)}, y^l_j)$$

definieren. Dabei gebe $\mu^l(k,r) = |\{j = 1,\dots,k : p^l(j) = r\}|$ *an, wie oft der Index* r *bisher durch den Pfad besucht wurde, und* $B > 1$ *sei ein Parameter. Die Übergangswahrscheinlichkeiten zu Index* r *werden bei dieser Definition mit jedem Besuch des Pfades durch Division durch* B *reduziert.*

4. *Eine Pfadkonstruktion mit redundanten Punkten, die nicht wie Algorithmus 2.83 an die adaptiv zufällige Pfadkonstruktion (Algorithmus 2.69), sondern an die adaptiv deterministische Pfadkonstruktion (Algorithmus 2.67) angelehnt ist, ist denkbar.*

Wir schlagen noch eine weniger komplexe, nicht adaptive Pfadkonstruktion vor.

Algorithmus 2.85 Nicht adaptive Pfadkonstruktion mit redundanten Punkten
Gegeben: $\Gamma^l = \{x_1^l,\dots,x_N^l\} \subset \mathbb{R}^d$.

1. *Wähle den ersten Index des Pfades* $p^l(1)$ *zufällig aus* $\{1,\dots,N\}$ *gemäß einer Gleichverteilung.*

2. *Durchlaufe folgende Iteration für* $k = 1,\dots,2N-1$*:*
   Nach der Konstruktion von $p^l(1),\dots,p^l(k)$ *im Pfadvektor setzen wir*

$$w(x_i^l, x_j^l) := \begin{cases} \frac{1}{B^{\mu^l(k,r)}} & \text{für } \|x_i^l - x_j^l\|_2 \le 2^{l/d} D_1 \\ 0 & \text{für } \|x_i^l - x_j^l\|_2 > 2^{l/d} D_1 \end{cases}, \tag{2.13}$$

   wobei $D_1 > 0$, $B > 1$ *Parameter sind und der Exponent* $\mu^l(k,r) = |\{j = 1,\dots,k : p^l(j) = r\}|$ *wieder angibt, wie oft der Index* r *bisher durch den Pfad besucht wurde. Wir berechnen*

$$s_{p^l(k)} = \sum_{j=1}^{N} w(x_{p^l(k)}^l, x_j^l)$$

   und bestimmen die Übergangswahrscheinlichkeiten $P_{p^l(k),r}$ *vom aktuellen Index* $p^l(k)$ *zu Index* r *vermöge*

$$P_{p^l(k),r} := \begin{cases} \frac{w(x_{p^l(k)}^l, x_r^l)}{s_{p^l(k)}} & \text{falls } s_{p^l(k)} \neq 0 \\ 0 & \text{sonst} \end{cases}. \tag{2.14}$$

   a) *Ist* $P_{p^l(k),r} \neq 0$ *für mindestens ein* $r \in \{1,\dots,N\}$*, so wähle* $p^l(k+1)$ *zufällig gemäß der eingeführten Wahrscheinlichkeitsverteilung.*

   b) *Ist* $P_{p^l(k),r} = 0$ *für alle* $r \in \{1,\dots,N\}$*, so wähle* $p^l(k+1)$ *zufällig gemäß einer Gleichverteilung unter den noch nicht verwendeten Indizes, d.h. aus*

$\{1,\ldots,N\} \setminus \cup_{j=1}^{k}\{p^l(j)\}$. *Falls bereits alle Indizes gewählt wurden, d.h. falls* $\{1,\ldots,N\} \setminus \cup_{j=1}^{k}\{p^l(j)\} = \emptyset$, *so wähle* $p^l(k+1)$ *zufällig gemäß einer Gleichverteilung unter allen Indizes* $\{1,\ldots,N\}$.

Ausgabe: *Pfadvektor* $p^l = (p^l(k))_{k=1}^{2N}$.

Bemerkung 2.86 *1. Algorithmus 2.85 berücksichtigt bei der Pfadkonstruktion die Tiefpasswerte* c_j^l *nicht und ist in diesem Sinne nicht adaptiv.*

2. Ein nachfolgender Punkt auf dem Pfad wird nach (2.13) *im Wesentlichen gemäß einer Gleichverteilung unter den spatialen Nachbarn des aktuellen Punktes* $x^l_{p^l(k)}$ *gewählt, wobei die Wahrscheinlichkeit für bereits besuchte Punkte heruntergesetzt wird.*

3. Der erwähnte Fall, dass $P_{p^l(k),r}$ *für alle* $r \in \{1,\ldots,N\}$ *verschwindet, tritt hier nur auf, wenn der aktuelle Punkt* $x^l_{p^l(k)}$ *keine spatialen Nachbarn besitzt. Durch passende Wahl des Parameters* D_1 *sollte diese Situation möglichst vermieden werden.*

In der Praxis konnten wir durch die Anwendung des Entstörungsalgorithmus mit redundanten Punkten (Algorithmus 2.81) trotz des erhöhten Aufwandes keine Verbesserung, sondern nur eine Verschlechterung der Entstörungsergebnisse des ursprünglichen Entstörungsalgorithmus (Algorithmus 2.63) realisieren. Man betrachte hierzu das Beispiel der Entstörung des „Cameraman"-Bildes (siehe Abbildung 2.19 und Tabelle 2.8). Die für dieses Beispiel gewählten Parameter sind in Tabelle 2.7 abzulesen. Dass einige Punkte bei der Pfadkonstruktion gegebenenfalls überhaupt nicht besucht werden, ist vermutlich der Hauptgrund, warum die Entstörung mittels des Ansatzes mit redundanten Punkten nicht leistungsfähiger ist. Auffällig ist, dass die weniger komplexe, nicht adaptive Pfadkonstruktion nach Algorithmus 2.85 sogar zu einem besseren Resultat als die adaptiv zufällige Pfadkonstruktion nach Algorithmus 2.83 führt. Es sei bemerkt, dass die erreichten PSNR-Werte und benötigten Laufzeiten bei mehrmaliger Anwendung des Algorithmus 2.81 mit redundanten Punkten stärker variieren als beim standardmäßigen Algorithmus 2.63. In diesem Sinne ist Algorithmus 2.81 weniger stabil als Algorithmus 2.63. Ein Erklärungsansatz hierfür ist, dass die Anzahl der Punkte, die gar nicht vom Pfad besucht werden, bei den unterschiedlichen Realisierungen der Pfadkonstruktion schwanken können.

Es sei noch auf die Möglichkeit, die Idee der Pfade mit redundanten Punkten mit dem Ansatz der einmaligen Pfadkonstruktion aus Unterabschnitt 2.8.1 zu kombinieren, hingewiesen. Man konstruiere hierzu einmalig einen Pfad $(p(k))_{k=1}^{2N}$ mit redundanten Punkten und wendet anschließend eine Wavelet-Filterbank mit mehreren Zerlegungsleveln auf den Vektor $\tilde{f}(x_{p(k)})_{k=1}^{2N}$ der verrauschten Funktionswerte entlang des Pfades an. Bei der Rekonstruktion verwende man analog zum Vorgehen in Algorithmus 2.81 für Punkte mit

nicht adaptive Pfadkonstruktion	4 Level, $D_1 = 1.3,\ \theta = 0.30,\ B = 100$
adaptiv zufällige Pfadkonstruktion	4 Level, $D_1 = 2,\ \theta = 0.30,\ \eta_1 = \eta_2 = 0.2$
einmalige, nicht adaptive Pfadkonstruktion	8 Level, $D_1 = 1.3,\ \theta = 0.28$
einmalige, adaptiv zufällige Pfadkonstruktion	6 Level, $D_1 = 2.5,\ \theta = 0.28,\ \eta_1 = \eta_2 = 0.1$

Tabelle 2.7: Verwendete Parameter für die Entstörung des „Cameraman“-Bildes ($\sigma = 0.1$) mittels Pfaden mit redundanten Punkten.

einer Vielfachheit größer als Eins das arithmetische Mittel der rekonstruierten Funktionswerte und für Punkte mit der Vielfachheit Null den ursprünglichen, ungeglätteten Funktionswert. Auf diese Weise verringert sich der Mehraufwand des Ansatzes mit redundanten Pfadpunkten. Im numerischen Test (siehe Abbildung 2.19 und Tabelle 2.8) ergaben sich bessere Ergebnisse als bei der Entstörung mit nicht redundanter, einmaliger Pfadkonstruktion (Algorithmus 2.75) und überraschend auch etwas bessere Ergebnisse gegenüber der Entstörung mit redundanter Pfadkonstruktion in jedem Level (Algorithmus 2.81). Für dieses Beispiel wurden die Parameter wie in Tabelle 2.7 angegeben gesetzt.

2.8.5 Verfahren mit Datenvorentstörung

Unsere Konstruktionen aus Abschnitt 2.4 zielen darauf ab, dass sich die Pfade an Strukturen innerhalb des Datensatzes anpassen. Durch die Störungen, mit denen die Funktionswerte $f(x_j)$ belegt sind, kann es dabei jedoch zu Fehlschlüssen kommen. Um die Adaption der Pfade an die Strukturen des Datensatzes robuster zu machen, haben wir - neben der an das Cycle-Spinning angelehnten Durchschnittsbildung über mehrere Iterationen - die Pfadkonstruktion der ursprünglichen EPWT modifiziert. Bei der adaptiv deterministischen Pfadkonstruktion (Algorithmus 2.67) haben wir hierzu ähnlich der relaxierten EPWT einen Toleranzbereich für die Differenz aufeinanderfolgender Funktionswerte, innerhalb dessen eine eingeschlagene Richtung des Pfades beibehalten wird, eingeführt. Die adaptiv zufällige Pfadkonstruktion (Algorithmus 2.69) wird durch die Einführung von Zufall robuster.

Möglicherweise sind diese Maßnahmen jedoch nicht weitgreifend genug. Die Idee ist nun, nach der Durchführung einer Vorentstörung verbesserte Pfade finden zu können. Man vergleiche das zweischrittige Vorgehen des BM3D-Algorithmus [36] (Grundidee in Abschnitt 2.7 beschrieben) zur Bildentstörung. Wir schlagen drei Varianten eines

Verfahrens mit Datenvorentstörung vor, wobei weitere ähnliche Ansätze denkbar sind. Um die Notation etwas zu verkürzen, bezeichne im Folgenden

$$\tilde{F} = (\tilde{f}(x_j))_{j=1}^N \in \mathbb{R}^N$$

den Vektor der gestörten Eingangsfunktionswerte und

$$\hat{F} = (\hat{f}(x_j))_{j=1}^N \in \mathbb{R}^N$$

die entsprechende Rekonstruktion der entstörten Funktionswerte.

Variante 1

Wir beginnen mit der einfachsten Variante eines Verfahrens mit Datenvorentstörung.

Algorithmus 2.87 *1. Wende Algorithmus 2.63 zur Vorentstörung auf $\tilde{F}$ an. Bilde dabei den Durchschnitt der Rekonstruktionen aus z.B. 16 Iterationen, um vorentstörte Funktionswerte $\hat{F}^0$ zu erhalten.*

2. Wende Algorithmus 2.63 auf $\tilde{F}$ an, wobei zur Pfadkonstruktion im ersten Level die vorentstörten Werte $\hat{F}^0$ verwendet werden. Bilde dabei den Durchschnitt der Rekonstruktionen aus z.B. 48 Iterationen, um die finale Rekonstruktion $\hat{F}$ zu erhalten.

Bemerkung 2.88 *1. Für den Spezialfall regulärer Bilddaten lässt sich im ersten Teilschritt natürlich auch ein anderer Entstörungsalgorithmus (z.B. Shrinkage der zweidimensionalen Tensorprodukt-Wavelet-Transformation mit Cycle-Spinning, siehe Algorithmus 2.59) zur Vorentstörung verwenden.*

2. Man beachte, dass im zweiten Teilschritt die Werte $\hat{F}^0$ nur zur Pfadkonstruktion und nur im ersten Level verwendet werden. Die Wavelet-Filter werden dann auf die ursprünglichen gestörten Daten $\tilde{F}$ entlang des konstruierten Pfades angewendet.

3. Die Anzahl der Iterationen der zwei Teilschritte wurde hier so vorgeschlagen, dass ihre Summe der Anzahl der Iterationen, welche wir bei der Anwendung des Algorithmus 2.63 üblicherweise wählen, entspricht.

Variante 2

Offensichtlicher Nachteil von Variante 1 ist, dass wir nur für das erste Level eine Verbesserung der Pfadkonstruktion durch die Verwendung der vorentstörten Daten erhoffen

können. Diesem Nachteil können wir begegnen, indem wir den Entstörungsalgorithmus 2.63 auch mit den vorentstörten Werten $\hat{F}^0$ durchlaufen und die resultierenden Tiefpass-Werte zur Pfadkonstruktion in den höheren Leveln nutzen.

Algorithmus 2.89 *1. Wende Algorithmus 2.63 zur Vorentstörung auf $\tilde{F}$ an. Bilde dabei den Durchschnitt der Rekonstruktionen aus z.B. 8 Iterationen, um vorentstörte Funktionswerte $\hat{F}^0$ zu erhalten.*

2. Führe Algorithmus 2.63 parallel für die gestörten und vorentstörten Funktionswerte $\tilde{F}$ und $\hat{F}^0$ durch. Nutze zur Pfadkonstruktion in beiden Fällen $\hat{F}^0$ (bzw. in höheren Leveln die aus $\hat{F}^0$ entstehenden Tiefpass-Werte). Bilde dabei den Durchschnitt der Rekonstruktionen aus z.B. 56 Iterationen. Die Ausgabe für die Behandlung von $\tilde{F}$ stellt die finale Rekonstruktion $\hat{F}$ der entstörten Funktionswerte dar.

Variante 3

Es ergibt sich die Frage, ob eine Unterteilung der Entstörung in genau zwei Schritte wie in Variante 1 und 2 optimal ist. Statt des zweischrittigen Vorgehens ist auch ein fließendes mehrschrittiges Vorgehen - wie im folgenden Algorithmus dargelegt - denkbar. Hierbei werden iterativ verbesserte Daten (in Form des Durchschnitts der bisherigen Iterationen als Zwischenergebnis) zur Pfadkonstruktion herangezogen.

Algorithmus 2.90 *1. Initialisiere $\hat{F}^0 = \tilde{F}$.*

2. Für $t = 1, \ldots, 64$:

- *Wende Algorithmus 2.63 mit einer Iteration auf $\tilde{F}$ an, wobei zur Pfadkonstruktion im ersten Level $\hat{F}^0$ verwendet wird, um entstörte Daten $\hat{F}^t$ zu erhalten.*
- *Aktualisiere $\hat{F}^0$ als Durchschnitt der bisherigen Iterationen*

$$\hat{F}^0 = \frac{1}{t} \sum_{s=1}^{t} \hat{F}^s.$$

3. Setze $\hat{F} = \hat{F}^0$ als die finale Rekonstruktion.

Bemerkung 2.91 *1. Möglich ist, bei der Aktualisierung von $\hat{F}^0$ nur den Durchschnitt der letzten z.B. 16 Entstörungsergebnisse - in der Hoffnung, dass diese aussagekräftiger sind - zu berücksichtigen.*

2. *Weiterhin ist denkbar, wie in Variante 2 in höheren Leveln für die Pfadkonstruktion die aus $\hat{F}^0$ entstehenden Tiefpass-Werte in Betracht zu ziehen.*

In der Praxis konnte leider keine der drei in diesem Unterabschnitt vorgestellten Varianten von Verfahren mit Datenvorentstörung eine wesentliche Verbesserung der Entstörungsergebnisse erbringen (siehe Abbildung 2.18 und Tabelle 2.8). Interessanterweise erzielten wir mit der Variante 1 ein besseres Resultat als mit der Variante 2. Für die dargestellten beispielhaften Resultate wurde die adaptiv deterministische Pfadkonstruktion verwendet. Die freien Parameter wurden mit Ausnahme des Threshold-Parameters θ bei Variante 2, welcher $\theta = 0.30$ gesetzt wurde, wie in Tabelle 2.5 gewählt.

2.8.6 Zufällig reduzierte Datensätze

Wir gehen noch auf einen Ansatz ein, welcher auf der Idee der Wavelet-Zerlegungen von Zufallswäldern (engl. wavelet decompositions of random forests) von Dekel und Nemirovsky für die in [39] diskutierten geometrischen Wavelets beruht. Die Grundidee hierzu stammt aus dem maschinellen Lernen und wird auch als „bagging“ (kurz für „bootstrap aggregating“) [23, 24] bezeichnet.

Wir adaptieren diesen Ansatz für unsere Methode, indem wir vor jeder Iteration des Entstörungsalgorithmus 2.63 zufällig einen gewissen Anteil der Datenpunkte $(x_j, \tilde{f}(x_j))^T$ ausschließen und den Algorithmus nur auf den zufällig reduzierten Datensatz anwenden. Hierdurch wird Zufall auf eine andere Art als bisher betrachtet in die Entstörung eingeführt. Die Durchschnittsbildung aus den Rekonstruktionen mehrerer Iterationen soll dabei wieder die Entstörungsgüte erhöhen. Nebenbei verringert sich der numerische Aufwand aufgrund des reduzierten Datensatzes leicht.

Für die abschließende Durchschnittsbildung der Ergebnisse der T Iterationen ergeben sich zwei Vorgehensweisen. Entweder bilden wir einen einfachen Durchschnitt, wobei wir einem Punkt für eine Iteration, in der er ausgeschlossen war, den originalen, ungeglätteten Funktionswert zuordnen, oder einen gewichteten Durchschnitt, bei dem wir berücksichtigen, in wie vielen Iterationen ein Punkt nicht ausgeschlossen war. In beiden Fällen, insbesondere im Fall der gewichteten Durchschnittsbildung, sollte die Anzahl T der Iterationen ausreichend hoch gewählt werden, sodass die Wahrscheinlichkeit, dass ein Punkt in jeder Iteration ausgeschlossen wird, möglichst gering ist.

Die Idee der zufällig reduzierten Datensätze hat sich für Entstörungsalgorithmus 2.63 in der Praxis als unzureichend erwiesen, wie sich beispielhaft in Abbildung 2.19 und Tabelle 2.8 erkennen lässt. Hierbei wurden vor jeder Iteration ca. 12.1% der Datenpunkte zufällig ausgeschlossen. Es wurde die adaptiv deterministische Pfadkonstruktion mit den Parametern $C_1 = 2$ und $\theta = 0.35$ sowie 6 Zerlegungsleveln für die Variante

gestörtes Bild	19.97
Algorithmus 2.63 (adaptiv deterministische Pfadkonstruktion)	28.28
einmalige Pfadkonstruktion	26.54
nicht adaptiv deterministische Pfadkonstruktion	26.86
Zerlegung des Datensatzes	28.26
redundante Punkte (nicht adaptive Pfadkonstruktion)	27.38
redundante Punkte (adaptiv zufällige Pfadkonstruktion)	27.24
einmalige, nicht adaptive Pfadkonstruktion mit redundanten Punkten	27.47
einmalige, adaptiv zufällige Pfadkonstruktion mit redundanten Punkten	27.43
Verfahren mit Datenvorentstörung, Variante 1	28.29
Verfahren mit Datenvorentstörung, Variante 2	28.09
Verfahren mit Datenvorentstörung, Variante 3	28.30
zufällig reduzierte Datensätze (einfacher Durchschnitt)	27.49
zufällig reduzierte Datensätze (gewichteter Durchschnitt)	27.79

Tabelle 2.8: Mittels der Modifikationen aus Abschnitt 2.8 erreichte PSNR-Werte bei der Entstörung des „Cameraman“-Bildes ($\sigma = 0.1$).

mit einfachem Durchschnitt bzw. 8 Zerlegungsleveln für die Variante mit gewichtetem Durchschnitt verwendet. Die Entstörungsqualität näherte sich in Experimenten der des unmodifizierten Algorithmus 2.63 von unten an, je weniger man den Datensatz reduzierte. Der mit der Reduktion des Datensatzes einhergehende Informationsverlust lässt sich offensichtlich nicht kompensieren.

Abbildung 2.18: Entstörung des „Cameraman“-Bildes ($\sigma = 0.1$) mittels der Modifikationen aus Abschnitt 2.8; obere Reihe: einmalige (links) und nicht adaptive (rechts) Pfadkonstruktion; mittlere Reihe: Zerlegung des Datensatzes (links) und Verfahren mit Datenvorentstörung, Variante 1 (rechts); untere Reihe: Verfahren mit Datenvorentstörung, Variante 2 (links) und 3 (rechts).

Abbildung 2.19: Entstörung des „Cameraman“-Bildes ($\sigma = 0.1$) mittels der Modifikationen aus Abschnitt 2.8; obere Reihe: redundante Punkte mit nicht adaptiver (links) und adaptiv zufälliger (rechts) Pfadkonstruktion; mittlere Reihe: Kombination von redundanten Punkten mit einmaliger, nicht adaptiver (links) und adaptiv zufälliger (rechts) Pfadkonstruktion; untere Reihe: zufällig reduzierte Datensätze mit einfachem (links) und gewichtetem (rechts) Durchschnitt.

Literaturverzeichnis

[1] *CurveLab*. http://www.curvelet.org/software.html, abgerufen am 04.09.2014

[2] *Matlab-Software für BM3D*. http://www.cs.tut.fi/~foi/GCF-BM3D/index.html#ref_software, abgerufen am 04.09.2014

[3] *Matlab-Software zur Easy-Path-Wavelet-Transformation*. http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software, abgerufen am 04.09.2014

[4] *ShearLab*. http://www.shearlab.org/index_software.html, abgerufen am 04.09.2014

[5] Antoniadis, A. ; Pham, D.: Wavelet regression for random or irregular design. In: *Computational Statistics and Data Analysis* 28 (1998), S. 353–360

[6] Arnoldi, W.: The principle of minimized iteration in the solution of the matrix eigenvalue problem. In: *Quarterly of Applied Mathematics* 9 (1951), S. 17–25

[7] Asaki, T. ; Vixie, K. ; Sottile, M. ; Cherepanov, P.: Image Denoising by Regularization on Characteristic Graphs. In: *Applied Mathematical Sciences* 4 (2010), S. 2541–2560

[8] Baker, C.: *The Numerical Treatment of Integral Equations*. Clarendon Press, 1977

[9] Ball, G. ; Hall, D.: A clustering technique for summarizing multivariate data. In: *Behavioral Science* 12 (1967), S. 153–155

[10] Baraniuk, R. ; Cohen, A. ; Wagner, R.: Approximation and compression of scattered data by meshless multiscale decompositions. In: *Applied and Computational Harmonic Analysis* 25 (2008), S. 133–147

[11] Bauer, H.-U. ; Villmann, T.: *Growing a hypercubical output space in a selforganizing feature map*. International Computer Science Institute, Berkeley - Technical Report TR-95-030, 1995

[12] Belkin, M. ; Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: *Advances in Neural Information Processing Systems (NIPS 2001)*, 2002

[13] Belkin, M. ; Niyogi, P.: Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. In: *Neural Computation* 15(6) (2003), S. 1373–1396

[14] Bell, A. ; Sejnowski, T.: An information maximization approach to blind separation and blind deconvolution. In: *Neural Computation* 7(6) (1995), S. 1129–1159

[15] Bellman, R.: *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961

[16] Bengio, Y. ; Paiement, J.-F. ; Vincent, P. ; Delalleau, O. ; Roux, N. L. ; Ouimet, M.: Out-of-sample-extensions for LLE, isomap, MDS, eigenmaps and spectral clustering. In: *Advances in Neural Information Processing Systems (NIPS 2003)*, 2004

[17] Bernstein, M. ; Silva, V. de ; Langford, J. ; Tenenbaum, J.: *Graph approximations to geodesics on embedded manifolds*. Stanford University, Palo Alto - Technical Report, 2000

[18] Bishop, C.: *Neural Networks for Pattern Recognition*. Oxford University Press, 1995

[19] Bishop, C. ; Svensen, M. ; Williams, C.: GTM: A principled alternative to the self-organizing map. In: *Advances in Neural Information Processing Systems (NIPS 1996)*, 1997, S. 354–360

[20] Bishop, C. ; Svensen, M. ; Williams, C.: GTM: A principled alternative to the self-organizing map. In: *Neural Computation* 10(1) (1998), S. 215–234

[21] Bossmann, F.: *Ein Algorithmus zur effizienten Berechnung von Nachbarschaften*. – Persönliche Kommunikation

[22] Brand, M.: Charting a manifold. In: *Advances in Neural Information Processing Systems (NIPS 2002)*, 2002, S. 985–992

[23] Breiman, L.: Random forests. In: *Machine Learning* 45 (2001), S. 5–32

[24] Breiman, L. ; Friedman, J. ; Stone, C. ; Olshen, R.: *Classification and Regression Trees*. Chapman and Hall/CRC, 1984

[25] Cai, T. ; Brown, L.: Wavelet shrinkage for nonequispaced samples. In: *Annals of Statistics* 26(5) (1998), S. 1783–1799

[26] Candes, E. ; Demanet, L.: The curvelet representation of wave propagators is optimally sparse. In: *Communications on Pure and Applied Mathematics* 58 (2004), S. 1472–1528

[27] Candes, E. ; Donoho, D.: Curvelets - a surprisingly effective nonadaptive representation for objects with edges. In: *Curves and Surface Fitting: Saint-Malo 1999*, 2000, S. 105–120

[28] Cayton, L.: *Algorithms for manifold learning.* University of California, San Diego - Technical Report CS2008-0923, 2005

[29] Chung, F.: *Spectral Graph Theory.* American Mathematical Society, 1997

[30] Cohen, A. ; Daubechies, I. ; Feauveau, J.: Biorthogonal bases of compactly supported wavelets. In: *Communications on Pure and Applied Mathematics* 45 (1992), S. 485–560

[31] Coifman, R. ; Donoho, D.: Translation-invariant de-noising. In: *Wavelets and Statistics*, 1995, S. 125–150

[32] Coifman, R. ; Lafon, S.: Diffusion maps. In: *Applied and Computational Harmonic Analysis* 21(1) (2006), S. 5–30

[33] Coifman, R. ; Lafon, S. ; Lee, A. ; Maggioni, M. ; Nadler, B. ; Warner, F. ; Zucker, S.: Geometric diffusion as a tool for harmonic analysis and structure definition of data, part i. In: *Proceedings of the National Academy of Sciences* 102(21) (2005), S. 7426–7431

[34] Coifman, R. ; Maggioni, M.: Diffusion Wavelets. In: *Applied and Computational Harmonic Analysis* 21(1) (2006), S. 53–94

[35] Dabov, K. ; Foi, A. ; Egiazarian, K.: Video denoising by sparse 3D transform-domain collaborative filtering. In: *Proceedings of the 15th European Signal Processing Conference, EUSIPCO 2007*, 2007

[36] Dabov, K. ; Foi, A. ; Katkovnik, V. ; Egiazarian, K.: Image denoising with block-matching and 3D filtering. In: *Proceedings of SPIE Electronic Imaging '06*, 2006

[37] Daubechies, I.: *Ten Lectures on Wavelets.* SIAM, 1992

[38] Dekel, S. ; Leviatan, D.: Adaptive Multivariate Approximation using Binary Space Partitions and Geometric Wavelets. In: *SIAM Journal on Numerical Analysis* 43 (2005), S. 707–732

[39] Dekel, S. ; Nemirovsky, I.: *Wavelet decompositions of Random Forests.* – Preprint, `http://www.shaidekel.com/#!papers/c13dn`, abgerufen am 04.09.2014

[40] Delouille, V.: *Nonparametric stochastic regression using design-adapted wavelets.* Université catholique de Louvain - Dissertation, 2002

[41] Demartines, P. ; Hérault, J.: Vector quantization and projection neural network. In: *Lecture Notes in Computer Science (volume 686)*, 1993, S. 328–333

[42] Demartines, P. ; Hérault, J.: CCA: Curvilinear component analysis. In: *15th Workshop GRETSI*, 1995

[43] Demartines, P. ; Hérault, J.: Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. In: *IEEE Transactions on Neural Networks* 8(1) (1997), S. 148–154

[44] DeMers, D. ; Cottrell, G.: Non-linear dimensionality reduction. In: *Advances in Neural Information Processing Systems (NIPS 1992)*, 1993, S. 580–587

[45] Dijkstra, E.: A note on two problems in connection with graphs. In: *Numerical Mathematics* 1 (1959), S. 269–271

[46] Donoho, D. ; Grimes, C.: Hessian eigenmaps: New locally linear embedding techniques for high-dimensional data. In: *Proceedings of the National Academy of Sciences* 102(21) (2005), S. 7426–7431

[47] Easley, G. ; Lim, W. ; Labate, D.: Sparse Directional Image Representations using the Discrete Shearlet Transform. In: *Applied and Computational Harmonic Analysis* 25 (2008), S. 25–46

[48] Ehler, M. ; Filbir, F. ; Mhaskar, H.: Locally Learning Biomedical Data Using Diffusion Frames. In: *Journal of Computational Biology* 19(11) (2012), S. 1251–1264

[49] Fokkema, D. ; Sleijpen, G. ; Vorst, H. van der: Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils. In: *SIAM Journal on Scientific Computing* 20(1) (1999), S. 94–125

[50] Forgy, E.: Cluster analysis of multivariate data: efficiency versus interpretability of classifications. In: *Biometrics* 21 (1965), S. 768

[51] Fritzke, B.: Growing cell structures - a self-organizing network for unsupervised and supervised learning. In: *Neural Networks* 7(9) (1994), S. 1441–1460

[52] Fritzke, B.: Growing grid - a self-organizing network with constant neighborhood range and adaptation strength. In: *Neural Processing Letters* 2(5) (1995), S. 9–13

[53] Gavish, M. ; Nadler, B. ; Coifman, R.: Multiscale wavelets on trees, graphs and high-dimensional data: Theory and applications to semi supervised learning. In: *Proceedings of the 27th International Conference on Machine Learning*, 2010

[54] Getreuer, P.: *Wavelet Transforms without the Wavelet Toolbox*. `http://www.getreuer.info/tutorials/matlabimaging`, abgerufen am 04.09.2014

[55] Goldberg, Y. ; Zakai, A. ; Kushnir, D. ; Ritov, Y.: Manifold learning: The price of normalization. In: *Journal of Machine Learning Research* 9 (2008), S. 1909–1939

[56] Gong, D. ; Sha, F. ; Medioni, G.: Locally linear denoising on image manifolds. In: *Proceedings of the 13th International Conference on Artifical Intelligence and Statistics*, 2010

[57] Göppert, J.: Topology-preserving interpolation in self-organizing maps. In: *Proceedings of NeuroNîmes 1993*, 1993, S. 425–434

[58] Guillemard, M. ; Heinen, D. ; Iske, A. ; Krause-Solberg, S. ; Plonka, G.: *Adaptive Approximation Algorithms for Sparse Data Representation.* – Preprint, `http://num.math.uni-goettingen.de/plonka/publications.html`, abgerufen am 04.09.2014

[59] Guo, K. ; Labate, D.: The construction of smooth Parseval frames of shearlets. In: *Mathematical Modelling of Natural Phenomena* 8(1) (2013), S. 82–105

[60] Haar, A.: Zur Theorie der orthogonalen Funktionensysteme. In: *Mathematische Annalen* 69 (1910), S. 331–371

[61] Hall, P. ; Turlach, B.: Interpolation methods for nonlinear wavelet regression with irregularly spaced design. In: *Annals of Statistics* 25(5) (1997), S. 1912–1925

[62] Ham, J. ; Lee, D. ; Mika, S. ; Schölkopf, B.: *A kernel view of the dimensionality reduction of manifolds.* Max-Planck-Institut für biologische Kybernetik, Tübingen - Technical Report TR-110, 2003

[63] Hammond, D. ; Vandergheynst, P. ; Gribonval, R.: Wavelets on Graphs via Spectral Graph Theory. In: *Applied Computational Harmonic Analysis* 30 (2011), S. 129–150

[64] Han, J. ; Moraga, C.: The influence of the sigmoid function parameters on the speed of backpropagation learning. In: *From Natural to Artificial Neural Computation*, 1995, S. 195–201

[65] Hein, M. ; Maier, M.: Manifold Denoising for finding natural representations of data. In: *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI-07)*, 2007, S. 1646–1649

[66] Heinen, D. ; Plonka, G.: Wavelet shrinkage on paths for denoising of scattered data. In: *Results in Mathematics* 62(3) (2012), S. 337–354

[67] Hérault, J. ; Jaussions-Picaud, C. ; Guérin-Dugué, A.: Curvilinear component analysis for high dimensional data representation: I. Theoretical aspects and practical use in the presence of noise. In: *Proceedings of IWANN'99*, 1999, S. 635–644

[68] Hinton, G.: Training products of experts by minimizing contrastive divergence. In: *Neural Computation* 14(8) (2002), S. 1771–1800

[69] Hinton, G. ; Osindero, S. ; Teh, Y.: A fast learning algorithm for deep belief nets. In: *Neural Computation* 18(7) (2006), S. 1527–1554

[70] Hinton, G. ; Roweis, S.: Stochastic neighbor embedding. In: *Advances in Neural Information Processing Systems (NIPS 2002)*, 2003, S. 833–840

[71] Hinton, G. ; Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. In: *Science* 313(5786) (2006), S. 504–507

[72] Hotelling, H.: Analysis of a complex of statistical variables into principal components. In: *Journal of Educational Psychology* 24 (1933), S. 417–441

[73] Jansen, M. ; Nason, G. ; Silverman, B.: Scattered data smoothing by empirical Bayesian shrinkage of second generation wavelet coefficients. In: *Wavelet Applications in Signal and Image Processing IX, Proceedings of SPIE (volume 4478)*, 2001, S. 87–97

[74] Karhunen, K.: Zur Spektraltheorie stochastischer Prozesse. In: *Annales Academiae Scientiarum Fennicae* 34 (1946)

[75] Kohonen, T.: Self-organization of topologically correct feature maps. In: *Biological Cybernetics* 43 (1982), S. 59–69

[76] Kovac, A. ; Silverman, B.: Extending the scope of wavelet regression methods by coefficient-dependent thresholding. In: *Journal of the American Statistical Association* 95 (2000), S. 172–183

[77] Kruskal, J.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. In: *Psychometrika* 29 (1964), S. 1–28

[78] Kryszkiewicz, M. ; Lasek, P.: TI-DBSCAN: Clustering with DBSCAN by Means of the Triangle Inequality. In: *Rough Sets and Current Trends in Computing - 7th International Conference, RSCTC 2010*, 2010, S. 60–69

[79] Labate, D. ; Lim, W. ; Kutyniok, G. ; Weiss, G.: Sparse multidimensional representation using shearlets. In: *Wavelets XI, SPIE Proceedings 5914*, 2005, S. 254–262

[80] Lafon, S. ; Lee, A.: Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning and data set parameterization. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(9) (2006), S. 1393–1403

[81] Lee, J. ; Archambeau, C. ; Verleysen, M.: Locally linear embedding versus Isotop. In: *Proceedings of ESANN 2003, 11th European Symposium on Artificial Neural Networks*, 2003, S. 527–534

[82] Lee, J. ; Lendasse, A. ; Donckers, N. ; Verleysen, M.: A robust nonlinear projection method. In: *Proceedings of ESANN 2000, 8th European Symposium on Artificial Neural Networks*, 2000, S. 13–20

[83] Lee, J. ; Lendasse, A. ; Verleysen, M.: Curvilinear distances analysis versus isomap. In: *Proceedings of ESANN 2002, 10th European Symposium on Artificial Neural Networks*, 2002, S. 185–192

[84] Lee, J. ; Verleysen, M. ; Jordan, M. (Hrsg.) ; Kleinberg, J. (Hrsg.) ; Schölkopf, B. (Hrsg.): *Nonlinear Dimensionality Reduction*. Springer, 2007

[85] Linde, Y. ; Buzo, A. ; Gray, R.: An algorithm for vector quantizer design. In: *IEEE Transactions on Communications* 28 (1980), S. 84–95

[86] Loève, M.: Fonctions aléatoire du second ordre. In: *Processus stochastiques et mouvement Brownien*, 1948, S. 299

[87] Ma, J. ; Plonka, G.: The curvelet transform: A review of recent applications. In: *IEEE Signal Processing Magazine* 27(2) (2010), S. 118–133

[88] Ma, J. ; Plonka, G. ; Chauris, H.: A new sparse representation of seismic data using adaptive easy-path wavelet transform. In: *IEEE Geoscience and Remote Sensing Letters* 7(3) (2010), S. 540–544

[89] Maaten, L. van der: *Matlab Toolbox for Dimensionality Reduction*. `http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html`, abgerufen am 04.09.2014

[90] Maaten, L. van der ; Hinton, G.: Visualizing data using t-SNE. In: *Journal of Machine Learning Research* 9(Nov) (2008), S. 2431–2456

[91] Maaten, L. van der ; Postma, E. ; Herik, H. van den: *Dimensionality Reduction: A Comparative Review*. Tilburg University - Technical Report TiCC TR 2009-005, 2009

[92] MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. Volume I: Statistics*, 1967, S. 281–297

[93] Mallat, S.: *A Wavelet Tour of Signal Processing*. Academic Press, 2008

[94] Mallat, S.: Geometrical Grouplets. In: *Applied and Computational Harmonic Analysis* 26(2) (2009), S. 161–180

[95] Malsburg, C. von der: Self-organization of orientation sensitive cells in the striate cortex. In: *Kybernetik* 8 (1973), S. 256–266

[96] Mrazek, P. ; Weickert, J. ; Steidl, G.: Correspondences between Wavelet Shrinkage and Nonlinear Diffusion. In: *Scale-Space Methods in Computer Vision*, 2003, S. 101–116

[97] Noel, G. ; Djouani, K. ; Hamam, Y.: Graph-based Image Sharpening Filter Applied to Image Denoising. In: *International Journal of Smart Home* 5(2) (2011)

[98] Pearson, K.: On lines and planes of closest fit to systems of points in space. In: *Philosophical Magazine* 2 (1901), S. 559–572

[99] Pekalska, E. ; Ridder, D. de ; Duin, R. ; Kraaijveld, M.: A new method of generalizing Sammon mapping with application to algorithm speed-up. In: *Proceedings of ASCI'99, 5th Annual Conference of the Advanced School for Computing and Imaging*, 1999, S. 221–228

[100] Peltonen, J. ; Klami, A. ; Kaski, S.: Learning metrics for information visualisation. In: *Proceedings of the 4th Workshop on Self-Organizing Maps (WSOM'03)*, 2003, S. 213–218

[101] Plonka, G.: The easy path wavelet transform: a new adaptive wavelet transform for sparse representation of two-dimensional data. In: *Multiscale Modeling and Simulation* 7 (2009), S. 1474–1496

[102] Plonka, G. ; Iske, A. ; Tenorth, S.: Optimal Representation of Piecewise Hölder Smooth Bivariate Functions by the Easy Path Wavelet Transform. In: *Journal of Approximation Theory* 176 (2013), S. 42–67

[103] Plonka, G. ; Rosca, D.: Sparse data representation on the sphere using the easy path wavelet transform. In: *Sampling Theory and Applications (SampTA'09)*, 2009, S. 255–258

[104] Plonka, G. ; Rosca, D.: Easy path wavelet transform on triangulations of the sphere. In: *Mathematical Geosciences* 42(7) (2010), S. 839–855

[105] Plonka, G. ; Tenorth, S. ; Iske, A.: Optimally Sparse Image Representation by the Easy Path Wavelet Transform. In: *International Journal of Wavelets, Multiresolution and Information Processing* 10(1) (2012), S. 1250007 (20 Seiten)

[106] Plonka, G. ; Tenorth, S. ; Rosca, D.: Image approximation by a hybrid method based on the easy path wavelet transform. In: *Asilomar'09 Proceedings of the 43rd Asilomar conference on signals, systems and computers*, 2009, S. 442–446

[107] Plonka, G. ; Tenorth, S. ; Rosca, D.: A hybrid method for image approximation using the easy path wavelet transform. In: *IEEE Transactions on Image Processing* 20(2) (2011), S. 372–381

[108] Ram, I. ; Elad, M. ; Cohen, I.: Generalized tree-based wavelet transform. In: *IEEE Transactions on Signal Processing* 59(9) (2011), S. 4199–4209

[109] Ram, I. ; Elad, M. ; Cohen, I.: Redundant Wavelets on Graphs and High Dimensional Data Clouds. In: *IEEE Signal Processing Letters* 19(5) (2012), S. 291–294

[110] Ram, I. ; Elad, M. ; Cohen, I.: Image Processing using Smooth Ordering of its Patches. In: *IEEE Transactions on Image Processing* 22(7) (2013), S. 2764–2774

[111] Ridder, D. de ; Duin, R.: Sammon's mapping using neural networks: A comparison. In: *Pattern Recognition Letters* 18(11-13) (1997), S. 1307–1316

[112] Ridder, D. de ; Franc, V.: Robust subspace mixture models using t-distributions. In: *Proceedings of the British Machine Vision Conference 2003*, 2003, S. 319–328

[113] Robbins, H. ; Monro, S.: A stochastic approximation method. In: *Annals of Mathematical Statistics* 22 (1951), S. 400–407

[114] Roweis, S. ; Saul, L.: Nonlinear dimensionality reduction by locally linear embedding. In: *Science* 290(5500) (2000), S. 2323–2326

[115] Rudin, L. ; Osher, S. ; Fatemi, E.: Nonlinear total variation based noise removal algorithms. In: *Physica D* 60 (1992), S. 259–268

[116] Sammon, J.: A nonlinear mapping algorithm for data structure analysis. In: *IEEE Transactions on Computers* CC-18(5) (1969), S. 401–409

[117] Saul, L. ; Roweis, S.: Think globally, fit locally: Unsupervised Learning of Nonlinear Manifolds. In: *Journal of Machine Learning Research* 4 (2003), S. 119–155

[118] Schölkopf, B. ; Smola, A. ; Müller, K.-R.: Nonlinear component analysis as a kernel eigenvalue problem. In: *Neural Computation* 10 (1998), S. 1299–1319

[119] Schumacher, H.: *Numerische Stabilität von Wavelet-Algorithmen.* Universität Rostock - Dissertation, 2003

[120] Sha, F. ; Saul, L.: Analysis and extension of spectral methods for nonlinear dimensionality reduction. In: *ICML 2005 - Proceedings of the 22nd International Conference on Machine Learning*, 2005, S. 785–792

[121] Shepard, R.: The analysis of proximities: Multidimensional scaling with an unknown distance function (parts 1 and 2). In: *Psychometrika* 27 (1962), S. 125–140, 219–249

[122] Shi, J. ; Malik, J.: Normalized cuts and image segmentation. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(8) (2000), S. 888–905

[123] Spearman, C.: General intelligence objectively determined and measured. In: *American Journal of Psychology* 15 (1904), S. 206–221

[124] Starck, J. ; Candes, E. ; D.Donoho: The curvelet transform for image denoising. In: *IEEE Transactions on Image Processing* 11 (2000), S. 670–684

[125] Svensen, M.: *GTM: The generative topographic mapping.* Aston University - Dissertation, 1998

[126] Sweldens, W.: The lifting scheme: A construction of second generation wavelets. In: *SIAM Journal on Mathematical Analysis* 29(2) (1997), S. 511–546

[127] Teh, Y. ; Roweis, S.: Automatic alignment of hidden representations. In: *Advances in Neural Information Processing Systems (NIPS 2002)*, 2002, S. 841–848

[128] Tenenbaum, J.: Mapping a manifold of perceptual observations. In: *Advances in Neural Information Processing Systems (NIPS 1997)*, 1998, S. 682–688

[129] Tenenbaum, J. ; Silva, V. de ; Langford, J.: A global geometric framework for nonlinear dimensionality reduction. In: *Science* 290(5500) (2000), S. 2319–2323

[130] Tenorth, S.: *Adaptive Waveletmethoden zur Approximation von Bildern.* Georg-August-Universität Göttingen - Dissertation, 2011

[131] Tipping, M. ; Bishop, C.: Mixtures of probabilistic principal component analysers. In: *Neural Computation* 11(2) (1999), S. 443–482

[132] Tomasi, C. ; Manduchi, R.: Bilateral filtering for gray and color images. In: *Proceedings of the 6th International Conference on Computer Vision*, 1998, S. 839–846

[133] Torgerson, W.: Multidimensional scaling, I: Theory and method. In: *Psychometrika* 17 (1952), S. 401–419

[134] Unnikrishnan, R. ; Hebert, M.: Denoising Manifold and Non-Manifold Point Clouds. In: *18th British Machine Vision Conference*, 2007

[135] Vandenberghe, L. ; Boyd, S.: Semidefinite programming. In: *SIAM Review* 38(1) (1996), S. 49–95

[136] Vanraes, E. ; Jansen, M. ; Butheel, A.: Stabilized wavelet transforms for non-equispaced data smoothing. In: *Signal Processing* 82(12) (2002), S. 1979–1990

[137] Venna, J. ; Kaski, S.: Local multidimensional scaling with controlled tradeoff between trustworthiness and continuity. In: *Proceedings of the 5th Workshop on Self-Organizing Maps (WSOM'05)*, 2005, S. 695–702

[138] Wang, B. ; Tu, Z.: Sparse Subspace Manifold Denoising. In: *The 26th Conference on Computer Vision and Pattern Recognition*, 2013

[139] Weinberger, K. ; Packer, B. ; Saul, L.: Unsupervised learning of image manifolds by semidefinite programming. In: *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005

[140] Weinberger, K. ; Saul, L.: An Introduction to Nonlinear Dimensionality Reduction by Maximum Variance Unfolding. In: *Proceedings of the National Conference on Artificial Intelligence*, 2006, S. 1683–1686

[141] Welk, M. ; Weickert, J. ; Steidl, G.: A four-pixel scheme for singular differential equations. In: *Scale-Space and PDE Methods in Computer Vision*, 2005, S. 610–621

[142] Yang, L.: Sammon's nonlinear mapping using geodesic distances. In: *Proceedings of the 17th International Conference on Pattern Recognition*, 2004, S. 303–306

[143] Young, G. ; Householder, A.: Discussion of a set of points in terms of their mutual distances. In: *Psychometrika* 3 (1938), S. 19–22

[144] Zhang, Z. ; Zha, H.: Principal manifolds and nonlinear dimensionality reduction via local tangent space alignment. In: *SIAM Journal of Scientific Computing* 26(1) (2004), S. 313–338

Lebenslauf

Dipl.-Math. Dennis Heinen

Adresse	Institut für Numerische und Angewandte Mathematik Universität Göttingen Lotzestraße 16-18 37083 Göttingen
Telefon:	+49 551 394515
E-Mail:	d.heinen@math.uni-goettingen.de
Homepage:	http://na.math.uni-goettingen.de/index.php?section=heinen

Persönliche Daten

Geburtsdatum	31. August 1985
Geburtsort	Duisburg

Akademische Ausbildung

seit 08/2010	Promotionsstudium an der Universität Göttingen, Betreuerin: Prof. G. Plonka-Hoch
04/2010-07/2010	Promotionsstipendium an der Universität Duisburg-Essen, Betreuerin: Prof. G. Plonka-Hoch
10/2005-03/2010	Mathematikstudium an der Universität Duisburg-Essen (Nebenfach Betriebswirtschaftslehre) 03/2010: Diplom in Mathematik (Note 1.1), Titel der Diplomarbeit: Bildapproximation mittels Wedgelets, Betreuerin: Prof. G. Plonka-Hoch
08/1996-06/2005	Reinhard-und-Max-Mannesmann-Gymnasium, Duisburg 06/2005: Abitur (Note 2.1)

Forschungstätigkeiten

seit 09/2011	Wissenschaftlicher Mitarbeiter am Institut für Numerische und Angewandte Mathematik, Universität Göttingen, im Rahmen des Projektes „Adaptive Approximation Algorithms for Sparse Data Representation“ des DFG-Schwerpunktprogrammes 1324 „Extraktion quantifizierbarer Information aus komplexen Systemen“

08/2010-07/2013	Kollegiat im DFG-Graduiertenkolleg 1023 „Identifikation in mathematischen Modellen“ am Institut für Numerische und Angewandte Mathematik, Universität Göttingen

Lehrtätigkeiten

Numerische Methoden der Signal- und Bildverarbeitung, Übungen, Sommersemester 2011, Institut für Numerische und Angewandte Mathematik, Universität Göttingen

Numerische Mathematik I, Übungen, Wintersemester 2010/2011, Institut für Numerische und Angewandte Mathematik, Universität Göttingen

Publikationen

Guillemard, M.; Heinen, D.; Iske, A.; Krause-Solberg, S.; Plonka, G.: Adaptive Approximation Algorithms for Sparse Data Representation. Preprint, `http://num.math.uni-goettingen.de/plonka/publications.html`, abgerufen am 04.09.2014

Heinen, D.; Plonka, G.: Wavelet shrinkage on paths for denoising of scattered data. In: *Results in Mathematics* 62(3) (2012), S. 337-354

Vorträge auf Konferenzen und Workshops

2014	„Mecklenburger Workshop Approximationsmethoden und schnelle Algorithmen“, 17.-20.03.2014 in Hasenwinkel
2013	„The Fourteenth International Conference on Applied Mathematics and Computer Science“, 29.-31.08.2013 in Cluj-Napoca (Rumänien)
	„23. Rhein-Ruhr-Workshop“, 01.-02.02.2013 in Bestwig
2012	„Jahrestreffen SPP 1324, 2012“, 28.-30.11.2012 in Eisenach
	„GAMM Jahrestagung 2012“, 26.-30.03.2012 in Darmstadt
2011	„GAMM Jahrestagung 2011“, 18.-21.04.2011 in Graz (Österreich)