GPGPU-BESCHLEUNIGTE REDUKTION BATHYMETRISCHER GELÄNDEMODELLE MIT T-SPLINE-FLÄCHEN

Von der Fakultät für Bauingenieurwesen und Geodäsie

der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades

DOKTOR-INGENIEUR

Dr.-Ing.

genehmigte Dissertation
von

Dipl.-Inf. Christian Asche

geboren am 19. Juni 1978, in Bochum

2016

Referent:	apl. Prof. Dr.-Ing. habil. Volker Berkhahn Gottfried Wilhelm Leibniz Universität Hannover
Korreferent:	Prof. Dr. Heinrich Müller Technische Universität Dortmund
Tag der Promotion:	24. Juni 2015

Zusammenfassung

Digitale bathymetrische Geländemodelle finden Verwendung bei Gewässerböden mit geringer geographischer Ausprägung. Deren Datendichte steht oftmals in keinem Verhältnis zu der betrachteten räumlichen Skala oder der Problemstellung. Diese Dissertation greift das Teilproblem der Datenreduktion auf und beschreibt einen Prozess, ein bathymetrisches Geländemodell in eine reduzierende Beschreibung als T-Netz zu überführen. T-Spline-Flächen stellen hierzu ein wegweisendes Werkzeug aus der jüngeren Theorie der geometrischen Freiformflächen bereit.

Kreuzungen aus wohl bekannten und neuartigen Lösungen decken die in dieser Arbeit durchgeführten Arbeitsschritte ab. Diese umfassen ausgewählte Aspekte der Bildverarbeitung, der räumlichen Datenstrukturen, der angepassten Flächenrückführung von T-Spline-Flächen und der Optimierung mit einem approximativen Dragging-Verfahren.

Wesentliche Arbeitsschritte der Reduktion verlaufen hochgradig parallel. GPGPU-Beschleunigung durch Grafikprozessoren bildet die technische Grundlage der Arbeit. Die zeitkritischen Operationen werden identifiziert und mit anwendungsspezifischen Datenstrukturen und Algorithmen auf Basis der OpenCL-Schnittstelle umgesetzt.

Schlagworte: T-Spline-Flächen, Flächenrückführung, GPGPU

Abstract

Digital bathymetric terrain models are used for soils with low geographical variability. Frequently these are in no relation to the observed spatial scale or the problem statement. This publication focusses on the sub-problem of data reduction and describes a process to convert a bathymetric terrain model into a reduced T-mesh description. Therefore T-spline surfaces provide a tool from the recent theory of geometric modelling.

Well researched and novel solutions cover substantial parts of the completed steps. They include aspects of image processing, spatial data structures, surface reconstruction of T-spline surfaces and an optimization based on a dragging approach.

Major steps of the reduction are highly parallel executable. This work is founded on GPGPU acceleration with graphics processors. The time-critical operations are identified and implemented with customized data structures and algorithms based on the OpenCL interface.

Keywords: T-spline surfaces, surface reconstruction, GPGPU

Selbständigkeitserklärung

Hiermit erkläre ich, die Dissertation selbst verfasst zu haben, keine Textabschnitte von Dritten oder eigener Prüfungsarbeiten ohne Kennzeichnung übernommen und alle benutzten Hilfsmittel und Quellen in meiner Arbeit angegeben zu haben.

Hannover, 10. Dezember 2014

Christian Asche

Inhaltsverzeichnis

Vorwort

In meiner Zeit als wissenschaftlicher Mitarbeiter an der Leibniz Universität Hannover habe ich Einblick in vielfältige und abwechslungsreiche Forschungsthemen erhalten. Mir wurde die dankenswerte Möglichkeit geboten, ein aktiver Teil der Forschungsgemeinschaft zu werden. Ich konnte mein Interesse an den aufgegriffenen Themen in der gesamten Breite verstärken, mir ein vertieftes Wissen aneignen und meine Kenntnisse für alle Beteiligten gewinnbringend umsetzen. Viele der daraus entstandenen Lösungen habe ich in Form meiner Dissertation zusammengetragen.

Der größte Dank geht an meine Frau und meine Familie für die ungebrochene Motivation und Hilfestellung in allen Lebenslagen. Meinen Kollegen am Institut für Bauinformatik danke ich für jeden weiteren Rat und die freundschaftliche Zeit und Unterstützung. Besonderer Dank geht an Herrn apl. Prof. Dr.-Ing. habil. Berkhahn und Herrn Prof. Dr. Müller für die lehrreiche und konstruktive Kritik an meiner Forschungsarbeit.

1 Einleitung

Im Zentrum der vorangegangenen Forschungstätigkeit steht eine Verbindung der Ingenieurwissenschaften mit der Informatik. Sehr gute Verknüpfungen ergeben sich aus der geometrischen Modellierung von Umweltdaten und der prozeduralen Verarbeitung dieser. Angetrieben von dieser Idee beschreibt diese Dissertation einen beispielhaften Prozess für die Reduktion von bathymetrischen Datenbeständen in das topologische Format von T-Netzen für dazugehörige T-Spline-Flächen und dessen Umsetzung mithilfe der GPGPU-Schnittstelle OpenCL.

1.1 Inhalt

Digitale bathymetrische Geländemodelle finden auch Verwendung bei Gewässerböden mit geringer geographischer Ausprägung. Deren Datendichte steht oftmals in keinem Verhältnis zu der betrachteten räumlichen Skala oder der Problemstellung. Zahlreiche Ingenieuranwendungen beruhen auf der Nutzung dieser Modelle und folglich auf der damit verbundenen, nicht unerheblichen zusätzlichen Komplexität bei der Erfassung und Verarbeitung der Daten. Diese Dissertation greift das Teilproblem der Datenreduktion auf und beschreibt einen Prozess, ein bathymetrisches Geländemodell in eine (bezogen auf die Datengröße) reduzierte Beschreibung als T-Netz zu überführen.

T-Spline-Flächen stellen ein wegweisendes Werkzeug aus der jüngeren Theorie zu geometrischen Freiformflächen bereit. Diese eignen sich besonders, um lokale Details unter Beachtung der parametrischen Stetigkeit genauer abzubilden. Im Zusammenhang mit der bathymetrischen Modellierung ergeben sich bislang unbeachtete Vorteile, da großräumige Flachwasser- oder Wattgebiete problemlos mit markanten Übergängen, wie z. B. Fahrrinnen, kombiniert werden.

Kreuzungen aus wohl bekannten und neuartigen bzw. neu interpretierten Lösungen decken die in dieser Arbeit durchgeführten Arbeitsschritte ab. Dies umfasst die im Folgenden genannten Aspekte:

- Eine direkte bzw. indirekte Herangehensweise kapselt den gesamten Prozess. Der indirekte Weg umfasst eine zusätzliche Vor- und Nachverarbeitung mit Operationen der Bildverarbeitung. Diese Kombination ist im Kontext bathymetrischer Geländemodelle neuartig.

- Ein Approximationsschema integriert eine Flächenrückführung, d. h. die Anpassung einer Freiformfläche an ein bekanntes bathymetrisches Geländemodell. Im Wechsel finden hierbei Schritte der Modifikation eines (T-)Kontrollpunktnetzes und der linearen Optimierung der Geometrie von Nicht-Standard-T-Spline-Flächen statt. Die Lösung unterscheidet sich von der bisher bekannten Praxis, die sich auf den einfachen Fall mit Standard-T-Spline-Flächen beschränkt.

- Das Dragging-Verfahren stellt eine alternative, progressive[1] Lösungsmethode für die Flächenrückführung dar und wird in dieser Arbeit für T-Spline-Flächen angepasst und umgesetzt.

[1] Im Sinne eines kontinuierlichen Fortschritts hinsichtlich eines Optimierungsziels.

- R-Bäume sind eine erprobte Datenstruktur in Geoinformationssystemen. Als erweiterte Datenstruktur finden diese hier Verwendung für effiziente, dynamische Operationen auf T-Netzen.
- Wesentliche Arbeitsschritte der Reduktion sind hochgradig parallel ausführbar. GPGPU-Beschleunigung[2] durch Grafikprozessoren bildet die technische Grundlage der Arbeit. Die zeitkritischen Operationen werden identifiziert und mit anwendungsspezifischen Datenstrukturen und Algorithmen auf Basis der OpenCL-Schnittstelle umgesetzt.

1.2 Aufbau der Arbeit

Der Aufbau der Dissertation folgt der Top-Down-Struktur des oben beschriebenen Prozesses für die Reduktion bathymetrischer Geländemodelle. Das Kapitel 2 („Stand der Forschung“) wirft einen Blick auf wichtige, dieser Arbeit zugrundeliegenden Forschungsgebiete und erläutert deren theoretische Hintergründe. Im folgenden Kapitel 3 („Reduktion bathymetrischer Geländemodelle“) werden die direkte und die indirekte Herangehensweise mit den zusätzlichen Operationen der Bildverarbeitung beschrieben. Kapitel 4 („Approximation und Flächenrückführung“) definiert ein modulares Approximationsschema und geht auf die Besonderheiten bei der Flächenrückführung mit Nicht-Standard-T-Spline-Flächen ein. In diesem Zusammenhang werden sowohl die lineare Optimierung und das Dragging-Verfahren umgesetzt. Im anschließenden Kapitel 5 („T-Netze und Operationen“) wird eine angepasste Datenstruktur für die benötigten T-Netze vorgestellt. Ein wesentlicher Teil der Arbeit beruht auf der OpenCL-Implementierung von Datenstrukturen und Algorithmen, diese sind in Kapitel 6 („GPGPU-Optimierung“) beschrieben. Die Arbeit schließt mit der Auswertung eines realen Beispiels für eine großflächige Bathymetrie in Kapitel 7 („Analyse“) ab.

1.3 Notation

Zur Vereinheitlichung der mathematischen Schreibweise gilt die folgende Notation. Diese ist angelehnt an [27]:

- $\mathbb{R}^d$ ist der d-dimensionale Raum der reellen Zahlen, $\mathbb{E}^d$ ist der euklidische Raum, $\mathbb{N}$ sind die natürlichen Zahlen.
- $\mathbf{X} \in \mathbb{R}^d$ bezeichnet einen d-dimensionalen Vektor.
- $\mathbf{p} \in \mathbb{R}^2$ oder $\mathbf{q} \in \mathbb{E}^3$ sind Punkte.
- $x \in \mathbb{R}$ bezeichnet einen reellen, skalaren Wert
- $f(\ldots) \to \mathbb{R}$ ist eine Funktion.
- $B^K(s)$, $N^K(s)$, $B_i^K(s)$ und $N_i^K(s)$ sind die Bernstein-Polynome in den B-Spline-Basisfunktionen.
- $B^K[\phi_{i_0}, \ldots, \phi_{i_{K+1}}](s)$ stellt die B-Spline-Basisfunktionen mit lokalen (horizontalen oder vertikalen) Knotenwerten ϕ_{i_0}, $\ldots$,$\phi_{i_{K+1}}$ am i-ten Kontrollpunkt dar.
- $\mathbf{t}(s,t) \in \mathbb{E}^3$ definiert eine Fläche über den Parameterwerten s und t.

[2] General Purpose Computation on Graphics Processing Unit

- $x(\mathbf{p})$ ist die Abbildung des dreidimensionalen Punkts $\mathbf{p}$ auf seine x-Komponente. Analog hierzu sind dies $y(\mathbf{p})$ und $z(\mathbf{p})$ für die y- und z-Komponente.
- M definiert eine Matrix.
- M^T bzw. $\mathbf{X}^T$ ist die Transponierte einer Matrix oder eines Vektors.

Darüber hinaus werden Quellcode-Objekte wie Variablen und Schlüsselwörter aus OpenCL zur besseren Unterscheidung beispielsweise durch `float4` ausgeschrieben.

1.4 Quellenangaben

Bei dem analysierten Modellgebiet handelt es sich um einen Ausschnitt an Laserscans und Peilfahrten des Fahrwassers im Weser-Außengebiet aus dem Jahr 2008. Die Daten der Bathymetrie stammen von der Wasser- und Schifffahrtsverwaltung des Bundes. Die Nutzung erfolgt gemäß der „Verordnung zur Festlegung der Nutzungsbestimmungen für die Bereitstellung von Geodaten des Bundes (GeoNutzV)“.

Teile der Arbeit verwenden freie Java-Software. *Scilab* ist ein an MATLAB angelehntes Softwarepaket für die numerische Mathematik mit einer API-Anbindung. Für die OpenCL-Integration wird *JavaCL* verwendet.

2 Stand der Forschung

Diese Arbeit beschreibt die effiziente Reduktion von bathymetrischen Messdaten mittels T-Spline-Freiformflächen. Insgesamt wird ein Prozess beschrieben, der verschiedene Aufgaben des Ingenieurwesens und der Informatik verbindet. Für ein grundlegendes Verständnis soll der Blick zunächst auf folgende ausgewählte Teilgebiete gerichtet sein:

- Digitale bathymetrische Geländemodelle
- Modellierung mit Spline-Flächen
- Approximation und Flächenrückführung
- GPGPU-Beschleunigung mit OpenCL.

2.1 Digitale Bathymetrische Geländemodelle

Digitale Geländemodelle stellen die Basis für vielfältige Anwendungen im Ingenieurwesen dar. Eine wesentliche Aufgabe ist die Kartografie und speziell die Erfassung und Verarbeitung von Gewässern und Gewässerböden durch bathymetrische Geländemodelle.

2.1.1 Bathymetrie

Als Bathymetrie bezeichnet man den Prozess der Erfassung und Kartografierung von Gewässerböden wie Meere und Wattgebiete, Seen oder Flüssen (Abbildung 2.1 und 2.2). Häufig sind derartige Gebiete durch eine hohe Strömungsaktivität geprägt, beispielsweise Flussdelta und -mündungen. Ein markantes Beispiel sind die norddeutschen Mündungsgebiete der Weser und der Elbe, da deren ausströmendes Wasser einerseits und aus gegenläufiger Richtung der Seegang und die meteorologischen Effekte andererseits einen starken Effekt auf die Wattenmeere und die nahe Küste ausüben [62]. Um die Sicherheit des Küstenbestands zu gewährleisten und den Schifffahrtsverkehr in diesen Gebieten nicht nachhaltig zu stören, sind fortwährende anthropogene Eingriffe wie der Bau von Deichen, das Errichten von Buhnen und das Ausbaggern der Fahrwasser nötig.

Seit Jahrhunderten existieren Karten der Meeresküsten und insbesondere der deutschen Nordseeküste. Neuzeitliche hydrografische Vermessungen liefern große Mengen an Geländedaten vom Meeresbett, der Küstenlinie und den nahen Binnengewässern. Die Daten vergangener Jahrzehnte stammen oft aus Echolotpeilungen von Messschiffen oder geodätischen Messungen. Laserscan-Überfliegungen oder Satellitendaten stellen den Stand der Technik dar. Als Folge dessen nimmt die räumliche und zeitliche Dichte der Messdaten enorm zu. Die Abstände der Messpunkte liegen oftmals im Dezimeter-Bereich oder darunter. Die großräumig auftretenden Merkmale eines Gewässers sind hierdurch deutlich überrepräsentiert.

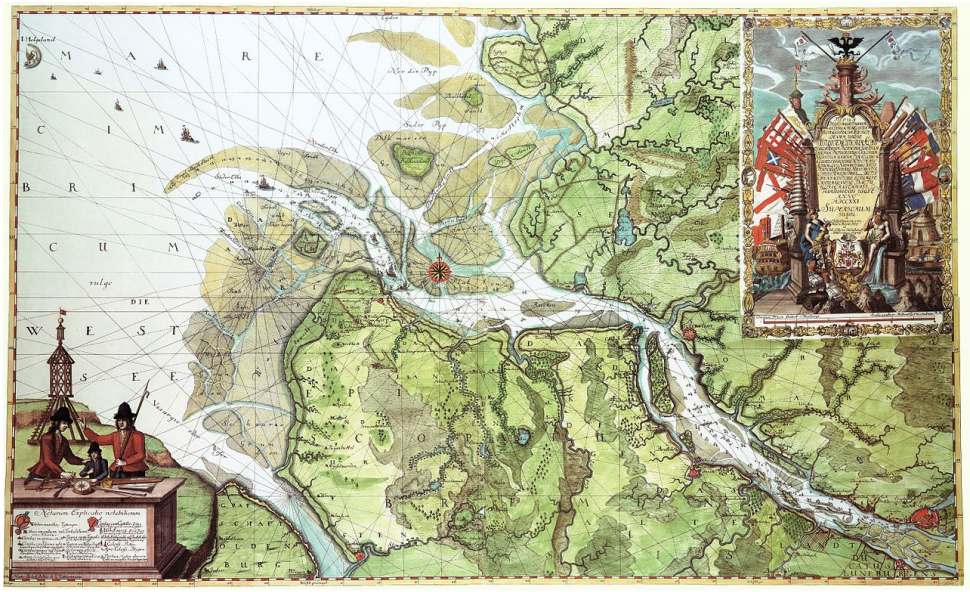


Abbildung 2.1: Karte der Elbmündung 1721 von Samuel Gottlieb Zimmermann und Otto Hasenbanck (Bildquelle: Wikimedia Commons, *public domain*)

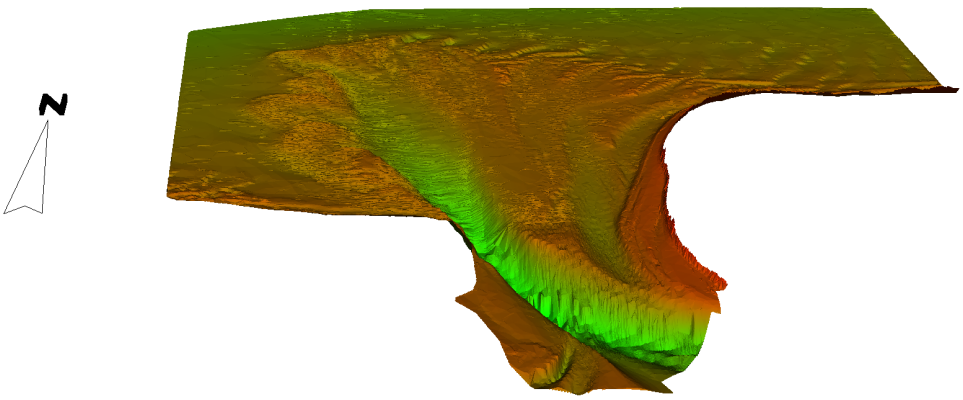


Abbildung 2.2: Bathymetrie für das Gebiet zwischen Langeoog und Spiekeroog (die positiven Höhenwerte der Landflächen sind ausgeblendet)

2.1.2 Hydrografie

Die Hydrografie befasst sich u. a. mit der Vermessung und nachfolgenden Kartografierung von Gewässern und Gewässerböden. Als solches umfassen wesentliche Arbeitsbereiche der Hydrografie die Erstellung und Verarbeitung von Bathymetrien. Hydrografische Karten basieren auf unterschiedlichen Messmethoden:

- Schiffsgebundene Sonarpeilungen
- Fernerkundung (Luftfahrt und Satelliten)
- Digitalisierung bestehender Karten
- Positionsbestimmung durch GPS[1].

Für die Erstellung von bathymetrischen Geländemodellen existiert eine Vielzahl an Sensortechniken, um den Meeresboden zu kartografieren. Diese unterscheiden sich wesentlich in ihrer horizontalen Messgenauigkeit, der vertikalen Durchdringung im Wasser, der simultanen flächenmäßigen Abdeckung, der hierfür benötigten Zeit und nicht zuletzt den dafür veranschlagten Kosten.

Hauptsächlich verwendet werden Fernerkundungsdaten von Satelliten und Fluggeräten, schiffsgebundene Sonarpeilungen und Videoaufzeichnungen für den direkten Nahbereich. Übliche Methoden der Fernerkundung umfassen die (Stereo-)Photogrammetrie, die Radargrammetrie und die Abtastung durch Laserstrahlen aus der Luft (*LIDAR*) [71]. Grundsätzlich werden die Messungen von Trägersystemen wie Schiffen, U-Booten oder Flugzeugen durch GPS-Ortsangaben aufbereitet. In Flachwassergebieten können die GPS-Höhendaten bei geringer Tide bereits ausreichend genaue Informationen liefern. Die Digitalisierung von historischem Kartenmaterial ergibt wahlweise eine Rastergrafik oder eine skalierbare Vektorzeichnung (z. B. für Konturlinien).

Eine Gegenüberstellung nach [44] und [55] führt die jeweilige horizontale Lagegenauigkeit, die Geschwindigkeit der Erfassung, den Aufwand / Kosten pro einzelnen Messpunkt und die abgedeckte Fläche ausgewählter Sensortechniken auf (Tabelle 2.1). Die Autoren ergänzen weitere Techniken, die für die Erstellung von Geländemodellen nicht geeignet sind, da sie im Wesentlichen für chemische und biologische Proben gedacht sind (z. B. Videodaten, Sedimentproben, o. ä.).

Sonarsysteme sind heutzutage der Stand der Technik für die Erstellung von Bathymetrien von tiefen Gewässern. Diese können in folgende Kategorien eingeteilt werden:

Single-Beam-Echolote senden einen einzelnen Sonarimpuls in vertikale Richtung. Die Tiefe wird anhand der Signallaufzeit bestimmt.

Fächer-Echolote verteilen mehrere aufgefächerte Sonarimpulse innerhalb eines gegebenen Öffnungswinkels. Die Tiefe wird ebenfalls über die Laufzeit bestimmt. Mit zunehmender Tiefe können breitere Gebiete erfasst werden, allerdings nimmt die horizontale Genauigkeit ab.

Flächen-Echolote ergänzen das Fächerecholot um Sonarimpulse mit variierenden Frequenzen und Phasen.

[1] *Global Positioning System*

Methode	**Lagegenauigkeit**	**Geschwindigkeit**	**Aufwand**	**Fläche**	**Anmerkung**
Sonar	$< 1\,\mathrm{m}$	langsam	sehr hoch	klein	
Satelliten-Photogrammetrie	$<100\,\mathrm{m}$	sehr schnell	gering	sehr groß	sehr hohe Initialkosten
Photogrammetrie	$<1\,\mathrm{m}$	schnell	sehr gering	mittel bis groß	nur in flachen Gewässern möglich
LIDAR	$<1\,\mathrm{m}$	schnell	mittel	mittel bis groß	nur in flachen Gewässern möglich
GPS	$<1\,\mathrm{m}$	langsam	sehr hoch	sehr klein	
Digitalisierung	$\sim 1\,\mathrm{m}$	langsam	hoch	beliebig	Genauigkeit abhängig vom Maßstab

Tabelle 2.1: Vergleich von Erhebungsmethoden für Digitale Geländemodelle nach [44] [55]

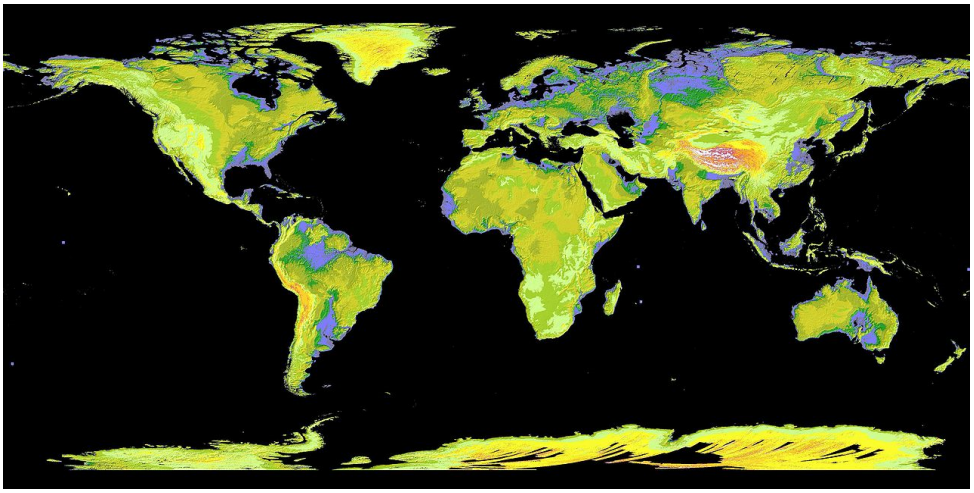


Abbildung 2.3: Digitales Höhenmodell der Erde erstellt aus der NASA-ASTER-Messung (*Advanced Spaceborne Thermal Emission and Reflection Radiometer*) (Bildquelle: Wikimedia Commons, *public domain*)

Sonarsysteme sind die hauptsächliche Datenquelle für tiefere Gewässer, allerdings nimmt die horizontale Messgenauigkeit mit zunehmender Tiefe deutlich ab. Fernerkundung kann leicht sehr große Gebiete abdecken, sie ist allerdings nur für seichte Wattenmeere geeignet. Die vertikale Durchdringung erreicht maximal nur wenige Meter und die Sensorik ist oftmals störanfällig für Effekte wie Lichtreflexion auf der Wasseroberfläche oder die Wassertemperatur. Nicht zuletzt erfassen Videodaten technisch bedingt nur kleinste Gebiete, können diese jedoch mit höchstmöglicher Auflösung abtasten und darüber hinaus weitere Informationen für die Klassifikation von Lebewesen und Sedimenten geben.

Im Unterschied zur Landvermessung weist die Hydrografie spezielle Eigenheiten auf. Neben der hohen zeitlichen Dynamik (es können geomorphodynamische Prozesse über mehrere Meter pro Jahr beobachtet werden) unterscheidet sie sich wesentlich durch die Notwendigkeit eines Tiefenwert-Bezugssystems mit einem definierten Nullpegel.

2.1.3 Digitale Geländemodelle

Ein Modell beschreibt ein reales oder gedachtes System. Hierfür werden die wichtigen Aspekte des Systems analysiert (Anwender, Aufgabe und Ziel) und in möglichst idealisierter und minimalisierter Form auf das Modell übertragen. Modelle bieten somit lediglich eine eingeschränkte Sicht auf die Wirklichkeit.

Ein Geländemodell stellt die Charakteristik eines geografischen Gebiets dar. Allgemein beinhaltet dies die Erfassung der Erdoberfläche bezogen auf ihre Höhen und Tiefen (Abbildung 2.3). Zusätzliche Bebauung oder Bepflanzung sind keine Aspekte der Modelle auf dieser Stufe. Ein Geländemodell umfasst ebenso keine weiteren Erd- oder Atmosphärenschichten. Die ersten antiken Geländemodelle bestehen aus gezeichneten Karten. Diese besitzen — aus heutiger Sicht — sowohl in ihrer räumlichen x-y-Position als auch in ihrer Höheninformation eine schlechte Genauigkeit. Piktogramme als Ersatz für Berge (Abbildung 2.1) und Konturlinien markieren extreme Landschaftsmerkmale.

Digitale Geländemodelle beschreiben Geländedaten mittels numerischer Rechenmodelle. Die Methoden der grafischen Verarbeitung bringen neue Möglichkeiten für die Erfassung, Bearbeitung und Darstellung von Digitalen Geländemodellen hervor.

Eine historische Einordnung nach Li et al. [55] ist:

Späte 1950er Einführung von Digitalen Geländemodellen in der Geodäsie

1960er Fotografische Fernerkundung

1960er bis 1970er Interpolationsmethoden und Sampling-Theorie

1980er Integration der Methoden der Bildverarbeitung

1990er Geo-Informationssysteme (GIS).

An Geländemodelle werden besondere Anforderungen gestellt:

- Genauigkeit / Korrektheit der Daten
- Realismus der Darstellung
- numerische Präzision
- Robustheit (bezüglich Datenausreißer)
- Verwendbarkeit
- Aussagekraft
- Einfachheit / Minimalismus.

Li et al. unterscheiden zwischen Gelände-, Höhen- und Erhöhungsmodellen [55]. Im englischen Sprachraum existieren die oft synonym verwendeten Begriffe *terrain model*, *elevation model*, *surface model* und *height model*.

Modellierung und Interpolation in Digitalen Geländemodellen

Eine grobe Unterteilung der Umsetzung von Digitalen Geländemodellen erfolgt zwischen mathematischen Modellen und grafischen Repräsentationen mit Punkten, Linien bzw. Dreiecken. Mathematische Modelle umfassen z. B. bilineare und bikubische Interpolation oder Radiale Basisfunktionen[2], hier speziell Multiquadriken nach Hardy.

Entscheidend für ein aussagekräftiges Modell innerhalb eines Anwendungsfalls ist oftmals die Verteilung der Daten: es gibt regulär verteilte Daten (z. B. aus der Fernerkundung) und irreguläre Daten. Letztgenannte sind in der Praxis häufig anzutreffen. Das sogenannte *Scattered Data Interpolation Problem* dient u. a. als Sammelbegriff für die damit verbundenen Probleme bezogen auf die Flächenrückführung (s. u.) [39]. Neben den unregelmäßig oder zufällig verteilten Messpunkten gibt es insbesondere bei Bathymetrien Spur- oder auch Profildaten (Abbildung 2.4). Diese stammen überwiegend aus Überfliegungen oder Echolotpeilungen. Hier bewegen sich die Messpunkte entlang einer Trajektorie durch das Vermessungsgebiet (z. B. bei Fahrrinnen), so dass die Daten insgesamt eine relativ hohe lokale Dichte aufweisen, bezogen auf ein Gesamtgebiet allerdings dünn ausfallen.

Eine Möglichkeit zur Erzeugung Digitaler Geländemodelle sind polynomielle Funktionen über einfache Basiselemente (Punkte, Dreiecke, Vierecke oder allgemeine Polygone). Eine weitere Unterscheidung ergibt sich aus der regulären oder unregelmäßigen Parkettierung der Basiselemente. Regelmäßig strukturierte Messdaten werden oftmals als

[2] Die Funktionswerte ergeben sich funktional über den Abstand zu einem Punkt. Der Abstand ist winkelunabhängig, daher die Bezeichnung „radial“.

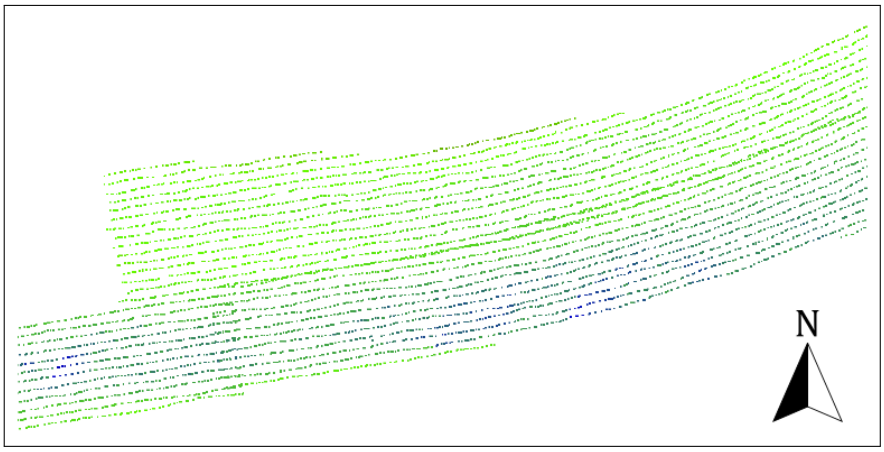


Abbildung 2.4: Hydrografische Spurdaten einer Peilfahrt

Gitter mit Punkten oder als unmittelbarer Rasterdatensatz modelliert. Ein solcher Rasterdatensatz besteht aus der Angabe von einem Ursprungspunkt und den x- bzw. y-Abständen der Rasterpunkte, so dass die Lage eines einzelnen Punkts direkt ermittelt wird. Die Effizienz basiert letztlich auf der deutlich reduzierten Speicherung der komprimierten Höhen- oder Tiefenwerte. Irreguläre Dreiecksnetze (*triangulated irregular network – TIN*) können beispielsweise über eine Delaunay-Triangulation der verteilten Messpunkte erzeugt werden. Effektiv hierfür ist der Bowyer-Watson-Algorithmus [13].

Fehlende Zwischenwerte in den Digitalen Geländemodellen werden schließlich über Interpolation berechnet. Aus der Literatur können vielzählige Verfahren für die flächendeckende Interpolation entnommen werden [58] [87] [69]. Üblich sind:

- bilineare Interpolation
- bikubische Interpolation
- inverse Distanzgewichtung
- Radiale Basisfunktionen.

Qualität und Fehler

Fehlerhafte Messungen oder falsche Konvertierungen können bereits in den Rohdaten auftreten. Der Fokus dieser Arbeit liegt allerdings auf der Erstellung eines bathymetrischen Geländemodells. Eine Qualitätsanalyse der Rohdaten ist bei den betrachteten Datensätzen nicht erforderlich und erfolgt somit nicht.

Bei Digitalen Geländemodellen gibt es zwei grundlegende Fehler: der horizontale x-y-Fehler und der vertikale z-Fehler. Ein gutes Fehlermaß für die Qualität eines Digitalen Geländemodells ist der quadratische Fehler E^2 als Differenzvolumen zwischen einer bathymetrischen Modellfläche $\mathbf{f}_{\text{bath}}(x, y)$ und einer zu berechnenden geometrischen Fläche $\mathbf{f}_{\text{geo}}(x, y)$ im dreidimensionalen Raum:

$$E^2 := \iint ||\mathbf{E}(x, y)||^2 \, dx \, dy \tag{2.1}$$

mit $\mathbf{E}(x, y) := \mathbf{f}_{\text{bath}}(x, y) - \mathbf{f}_{\text{geo}}(x, y)$. Die Norm $||\mathbf{E}(x, y)||^2$ beschreibt hierbei den Fehler über den ermittelten Differenzvektor (vgl. Abschnitt 2.3.2).

Folgende Aspekte beeinflussen die Genauigkeit eines Digitalen Geländemodells wesentlich:

- Die Struktur des abzubildenden Geländes: hier zeigen sich deutliche Unterschiede zwischen Gebieten mit hoch und gering ausgeprägten Landschaftsmerkmalen wie Plateaus oder Kanten.
- Die eingesetzte numerische Modellierung und die Wahl der damit verbundenen Interpolationsmethode: es zeigt sich ein Zusammenhang zwischen der Geländestruktur und der umgesetzten Interpolation, so dass beispielsweise Gebiete mit flacher Ausprägung sehr gut durch polynomielle Basisfunktionen erfasst werden können.
- Die Charakteristik der eingesetzten numerischen Interpolation: die Wahl der Parametrisierung des Modells hat deutlichen Einfluss auf die Qualität des Geländemodells. Höhergradige polynomielle Basisfunktionen eigenen sich meist nicht für Gelände mit harten Kanten und Übergängen.

Auf den davon unabhängigen Einfluss der unterschiedlichen Messverfahren, bezogen auf die Genauigkeit, Dichte und Verteilung der primären Messdaten, wird bereits im vorherigen Abschnitt eingegangen. Gute Modellbeschreibungen sind somit in der Lage, charakteristische Merkmale, abhängig vom Zweck des Modells, in einer Landschaft zu erfassen und qualitativ hochwertig zu modellieren. Zu diesen Merkmalen gehören z. B. Klippen, Plateaus, Risse, Konturlinien oder Vorsprünge. Letztere sind insofern eine Besonderheit, da hier keine funktionale Abbildung von einer Geokoordinate zu einem eindeutigen Höhenwert erfolgt.

Anwendungen von Digitalen Geländemodellen

Nach Li et al. finden sich Digitale Geländemodelle in vielen Teilgebieten des Bauwesens [55]. Ein Beispiel hierfür ist die Streckenplanung für Straße oder Schiene. Eine effektive Trassenfindung beruht auf der Minimierung von horizontaler Krümmung und vertikaler Gradienten innerhalb der Trassierungselemente. Ein weiteres Anwendungsgebiet ist die Planung von Wasserstrecken, die besonders die abwärtsgerichtete Flussrichtung des Wassers berücksichtigt.

Bathymetrien sind in der marinen Navigation zwingend notwendig. Nur so können ausreichend tiefe Fahrrinnen identifiziert und gegebenenfalls morphologisch aktive Gebiete bestimmt werden [62].

Weitere beispielhafte Anwendungen finden Digitale Geländemodelle in der Simulation von hydrodynamischen Vorgängen (z. B. Überflutung). Nicht zuletzt finden sich reduzierte Geländemodelle in der Darstellung von virtuellen Landschaften für zivile / militärische Simulationen und in Computerspielen wieder.

2.1.4 Forschungsbedarf

Hydrografische Messungen erzeugen verteilte Tiefenwerte als eine Punktwolke. Diese Daten haben einerseits eine hohe räumliche Abdeckung (z. T. über mehrere Quadratkilometer), andererseits jedoch lokal unterschiedliche Dichten. Die Ungenauigkeiten ergeben sich bereits aus der Messmethode (die von Fächerecholoten gleichzeitig erfassten Gebiete nehmen mit zunehmender Wassertiefe zu, die einzelnen Messpunkte weisen jedoch größere Abstände zueinander auf), andererseits aus der fehlerbehafteten Positionsbestimmung der Messsysteme. Nicht zuletzt nimmt die Datendichte durch

den Start von Satellitenmissionen und neuartigen Sensoren deutlich zu. Bereits heute werden Datendichten unterhalb von 1 m Punktabstand erreicht, allerdings steigt die benötigte Genauigkeit nicht in jedem Fall gleich an. Einzelnen Teilgebieten mit geringer morphologischer Dynamik stehen oftmals hochaktive Bereiche wie Fahrrinnen oder Küstenschutzmaßnahmen entgegen. Daher ist es ist schwierig, eine allumfassende Modellierung zu finden. Bathymetriegestützte Simulationen sind mit der puren Datenmenge somit schlicht überfordert, so dass vorab oftmals eine Datenreduktion erfolgt [100].

Die Idee, hochaufgelöste bathymetrische Rasterdaten durch Methoden der Wavelet-Bildkompression zu reduzieren, ist in [5] beschrieben. Allerdings besteht hierbei weiterhin der wesentliche Nachteil, dass ein Messgebiet stets rechteckig modelliert wird, so dass unwichtige Nebengebiete, Festland oder Inseln als Teil der Bathymetrie erhalten bleiben.
Die zu beachtenden Aspekte hydrografischer Messungen sind daher

1. die massive Datenmenge, und

2. die lokal ungleiche Dichte / Abstände der Messpunkte.

Der Fokus dieser Arbeit liegt auf einer Modellierung, die eine wesentliche Reduktion von verteilten hydrografischen Messdaten bewirkt. Hierfür werden T-Netze konstruiert, die eine vereinfachte und (bezogen auf die Anzahl der Kontrollpunkte) reduzierte Modellierung von Bathymetrien bezwecken. Eine Lösung für die Erstellung eines bathymetrischen Geländemodells bilden Freiformflächen. Ein Gelände wird hierbei durch eine parametrische Fläche im dreidimensionalen Raum modelliert. Im folgenden Abschnitt werden die Spline-Modelle vorgestellt, die die mathematischen Grundlagen der verwendeten Flächenbeschreibungen bilden.

2.2 Geometrische Modellierung mit Spline-Flächen

Die geometrische Modellierung erfasst die mathematische Beschreibung von Freiformkurven und -flächen. Rationale Spline-Flächen, und speziell T-Spline-Flächen, sind ein relevantes Teilgebiet.

2.2.1 Parametrische Flächen

Eine parametrische Fläche $\mathbf{x}(s,t) \in \mathbb{E}^3$ bildet jedes Parameterpaar $(s,t) \in \mathbb{R}^2$ auf einen Punkt $\mathbf{x} \in \mathbb{E}^3$ ab. $\mathbf{x}(s,t)$ ist komponentenweise definiert, und somit gilt $\mathbf{x}(s,t) := (x(s,t), y(s,t), z(s,t))$ [66]. Hierbei sind $x(s,t)$, $y(s,t)$ und $z(s,t)$ die getrennten Projektionen von $\mathbf{x}$ auf die x-, y- und z-Koordinaten.

Kontrollpunktnetz

Parametrische Flächen verwenden ein Netz aus Kontrollpunkten für die Modellierung einer Freiformfläche. Im CAD sind vollbesetzte Gitter (als spezielle Netztopologie) üblich. Diese bestehen aus den Reihenpolygonen P_i (mit $i = 0 \dots n-1$) und den Spaltenpolygonen Q_j (mit $j = 0 \dots m-1$). An der Kreuzung jedes Reihen- und Spaltenpolygons liegt ein Kontrollpunkt $\mathbf{p_{i,j}}$ (Abbildung 2.5).

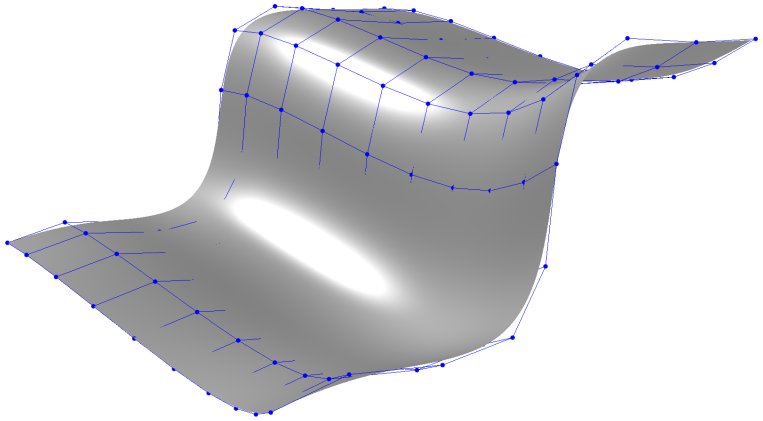


Abbildung 2.5: Spline-Fläche mit Kontrollpunktnetz

2.2.2 Spline-Flächen

Lokale Modellierbarkeit bezeichnet die Eigenschaft einer Fläche, bei der eine Änderung an einem Kontrollpunkt lediglich einen beschränkten Einfluss auf den Verlauf der Fläche nimmt. Eine Möglichkeit, lokale Modellierbarkeit umzusetzen, besteht in der Verwendung von Spline-Flächen.

Derartige Flächen bestehen aus Flächensegmenten, die eine einzelne stetige Fläche bilden. Hierfür werden zusätzlich Knotenwerte verwendet (s. u.), die die Intervallgrenzen der Parameterwerte der einzelnen Flächensegmente definieren.

B-Spline-Flächen

B-Spline-Flächen sind definiert als [27]

$$\mathbf{b}(s,t) := \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \mathbf{p_{i,j}} \cdot B_i^K(s) \cdot B_j^K(t) \tag{2.2}$$

mit den rekursiven B-Spline-Basisfunktionen

$$B_i^r(s) := \frac{s - \phi_i}{\phi_{i+r} - \phi_i} \cdot B_i^{r-1}(s) + \frac{\phi_{i+r+1} - s}{\phi_{i+r+1} - \phi_{i+1}} \cdot B_{i-1}^{r-1}(s) \tag{2.3}$$

und

$$B_i^0(s) := \begin{cases} 1 \text{ für } s \in [\phi_i, \phi_{i+1}[\\ 0 \text{ sonst.} \end{cases} \tag{2.4}$$

Dabei sind

- $\mathbf{b}(s,t)$ der Punkt auf der B-Spline-Fläche für den Parameterwert (s,t).
- $\mathbf{p_{i,j}} \in \mathbb{E}^3$ die geometrische Koordinaten des Kontrollpunkts im i-ten Reihen- und j-ten Spaltenpolygon des Kontrollpunktgitters.
- m und n die Anzahl der Reihen- und Spaltenpolygone im Kontrollpunktgitter.
- K der (Ansatz-)Grad der Fläche[3].

[3] Hier ist eine zusätzliche Unterscheidung von horizontalem und vertikalem Grad möglich.

- $\phi := [\phi_0, \ldots, \phi_{K+m+1}]$ die globalen, monoton steigenden Knotenwerte[4] (*knot values*). Horizontale Knotenwerte werden bei zweidimensionalen Flächen üblicherweise mit σ und vertikale Knotenwerte mit τ bezeichnet.

- $B_i^r(s)$ die B-Spline-Basisfunktion für das i-te Reihenpolygon der Kontrollpunkte (für die Spaltenpolygone analog).

2.2.3 NURBS-Flächen

Nicht-uniforme, rationale B-Spline-Flächen (*NURBS*) sind eine Verallgemeinerung der B-Spline-Flächen (Formel (2.2)) [27]. Gewichtete Kontrollpunkte mit weiterhin frei wählbaren (d. h. nicht-uniformen) Knotenwerten bereichern die Modellierung von Spline-Flächen um einen weiteren Freiheitsgrad.

$w_{i,j} \geq 0 \in \mathbb{R}$ bezeichnet ein Gewicht, das dem Kontrollpunkt $\mathbf{p_{i,j}}$ zugeordnet wird. Dies verletzt ggf. die Bedingung der affinen Kombination in den B-Spline-Basisfunktionen, d. h. die Summe der Ansatzfunktionen ist $\neq 1$. Die Erweiterung der Funktion (2.2) um einen rationalen Term korrigiert dies.
Somit gilt:

$$\mathbf{b}(s,t) := \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} w_{i,j} \cdot \mathbf{p_{i,j}} \cdot B_i^K(s) \cdot B_j^K(t)}{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} w_{i,j} \cdot B_i^K(s) \cdot B_j^K(t)} \tag{2.5}$$

2.2.4 Stetigkeit

Zwei Flächen $\mathbf{x}(s,t)$ und $\mathbf{y}(u,v)$ sind an einem gemeinsamen Punkt $\mathbf{x}(\hat{s},\hat{t})$ bzw. $\mathbf{y}(\hat{u},\hat{v})$ parametrisch C^k-stetig, wenn die ersten k partiellen Ableitungen von $\mathbf{x}$ und $\mathbf{y}$ stetig sind, d. h.

$$\frac{\partial^r \mathbf{x}(\hat{s},\hat{t})}{\partial s^i \partial t^j} = \frac{\partial^r \mathbf{y}(\hat{u},\hat{v})}{\partial u^i \partial v^j} \tag{2.6}$$

für $i,j > 0$, $i + j = r$ und $r = 0, \ldots, k$.

Die Verwendung von parametrisch stetigen Flächen ist wesentlicher Bestandteil dieser Arbeit im Zusammenhang mit dem betrachteten Forschungsgebiet.

2.2.5 T-Spline-Flächen

Sederberg et al. haben 2003 erstmalig die T-Spline-Flächen als eine Verallgemeinerung von NURBS-Flächen eingeführt [77]. Eine T-Spline-Fläche verwendet, im Unterschied zu NURBS-Flächen, lokale Knotenwerte. Diese sind nicht global definiert, sondern werden für jeden Knoten aus einem sogenannten T-Netz hergeleitet (s. u.). Diverse Arbeiten haben sich im Anschluss mit der Modellierung und der Anwendung von T-Spline-Flächen beschäftigt, einen Überblick gibt [3].

[4] Oftmals auch als *Knotenvektor* bezeichnet

Definition

Eine T-Spline-Fläche erweitert eine NURBS-Fläche um die Verwendung von lokalen Knotenwerten für eine (eindimensional indizierbare) Menge an Kontrollpunkten. Die Definition von T-Spline-Flächen lautet:

$$\mathbf{t}(s,t) := \frac{\sum_{i=0}^{n-1} w_i \cdot \mathbf{p_i} \cdot N^K[\sigma_i, \tau_i](s,t)}{\sum_{i=0}^{n-1} w_i \cdot N^K[\sigma_i, \tau_i](s,t)} \tag{2.7}$$

und unter Verwendung der B-Spline-Basisfunktionen (2.3) sei

$$N^K[\sigma_i, \tau_i](s,t) := B^K[\sigma_{i_0}, \ldots, \sigma_{i_{K+1}}](s) \cdot B^K[\tau_{i_0}, \ldots, \tau_{i_{K+1}}](t). \tag{2.8}$$

Hierbei ist K der Grad der Basisfunktion und $\mathbf{p_i}$ ein Kontrollpunkt mit Gewicht w_i. Die Notation $B^K[\phi_{i_0}, \ldots, \phi_{i_{K+1}}](s)$ beschreibt die B-Spline-Basisfunktionen (2.3) mit lokalen (horizontalen oder vertikalen) Knotenwerten $[\phi_{i_0}, \ldots, \phi_{i_{K+1}}]$ am i-ten Kontrollpunkt. Der Index i aus der B-Spline-Basisfunktion entfällt in $N^K[\ldots](\ldots)$. Stattdessen beziehen sich die lokalen Knotenwerte $\sigma_{i,*}$ und $\tau_{i,*}$ auf den i-ten Kontrollpunkt. Die Methodik zur Herleitung dieser Knotenwerte folgt in Abschnitt 5.2.

T-Netze

T-Spline-Flächen sind auf T-Netzen definiert, die T-Punkte (d. h. Kontrollpunkte mit drei angrenzenden Kanten) zulassen und somit die Benennung motivieren. Die eigentlichen T-Netze bestehen dabei aus eindimensional indizierten Kontrollpunkten, die durch Kanten verbunden werden. (Rechteckige) Zellen wiederum sind eine Verbindung von Kanten und bilden in dieser Arbeit die wesentlichen Elemente im T-Netz.

Die Konstruktion eines T-Netzes erfolgt durch zwei gegensätzliche Herangehensweisen. Zum einen ist ein T-Netz ein Kontrollpunktnetz, in dem Kanten aus einem Gitter, d. h. Verbindungen zwischen den Kontrollpunkten, fehlen dürfen. Auf diese Weise entstehen innere, d. h. nicht am Rand liegende, Kontrollpunkte mit weniger als vier inzidenten Kanten, sogenannte T-Punkte oder auch T-Kreuzungen. Andererseits können T-Netze stückweise aus der Verbindung von Kontrollpunkten durch Kanten erzeugt werden. Die Anzahl der Kanten an einem Kontrollpunkt darf dabei auch für innere Kontrollpunkte kleiner als vier sein. Abbildung 2.6 zeigt ein T-Netz und die daraus generierte T-Spline-Fläche. T-Netze werden detailliert in Kapitel 5 erläutert.

Nur wenige Publikationen und Arbeiten betrachten die T-Netze gesondert. Oftmals fehlt eine konkrete Beschreibung der Umsetzung ihrer Modellierung oder der eingesetzten Datenstrukturen. Es gibt keinen allgemeinen Konsens über eine Modellierung von T-Netzen, da diese in ihren Anwendungsfällen selten statisch sind, sondern zumeist dynamisch durch Prozesse verarbeitet werden. Hieraus ergeben sich die unterschiedlichen, anwendungsbezogenen Modelle. Weitere Ideen für die Implementierung von T-Netzen, basierend auf den Relationen zwischen Flächen, Kanten und Knoten, finden sich z. B. in [56] [6] [21].

Point-Based-Spline-Flächen

Sederberg et al. beschreiben die sogenannten „*point-based splines*“ (*PB splines*) als netzlose T-Spline-Flächen [77]. Weiterhin werden Kontrollpunkte mit dedizierten

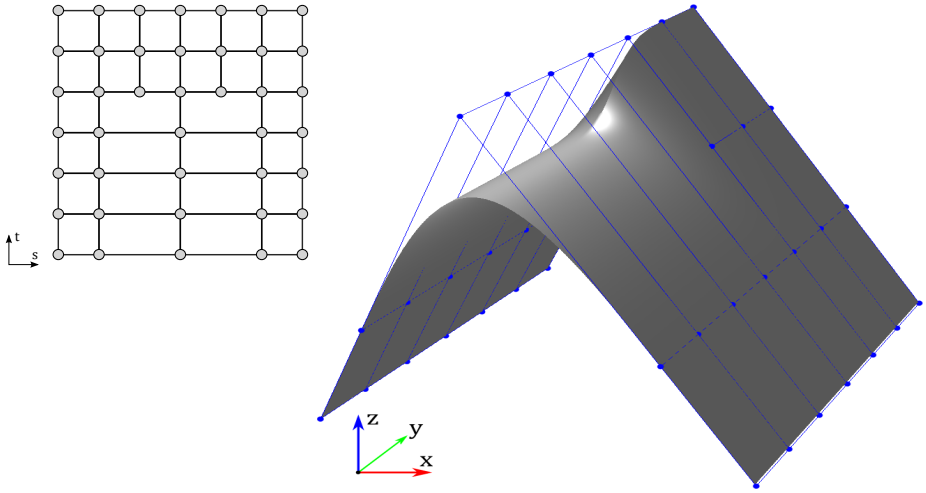


Abbildung 2.6: Darstellung eines T-Netzes im Parameterraum (links) und die generierte T-Spline-Fläche (rechts)

lokalen Knotenwerten vorgegeben, allerdings besitzt das T-Netz selbst keine weitere Bedeutung mehr. Die T-Spline-Funktion (2.7) bleibt gleich. Eine T-Spline-Fläche kann somit auf die Darstellung einer PB-Spline-Fläche mit identischen Kontrollpunkten und Knotenwerten reduziert werden (vgl. Abschnitt 6.2.2).

2.2.6 Eigenschaften der T-Spline-Flächen

NURBS- und T-Spline-Flächen beruhen auf den gleichen Basisfunktionen. Daher besitzen bereits NURBS-Flächen die folgenden Eigenschaften, die von T-Spline-Flächen größtenteils übernommen werden [66] [1]:

- Es gilt die Nicht-Negativität der Basisfunktion.
- Es gilt die Partition der Eins (*partition of unity*) in den Basisfunktionen.
- Der lokal beschränkte Einfluss der Basisfunktionen unterstützt (neben der Verfeinerung) die lokale Verarbeitung von bathymetrischen Details (s. u.).
- Affine Transformationen am Kontrollpunktnetz führen zu affinen Transformationen der erzeugten Fläche. Eine Fläche mit dieser Eigenschaft wird als affin invariant bezeichnet.
- Die erzeugte Fläche liegt innerhalb der konvexen Hülle ihres Kontrollpunktgitters bzw. T-Netzes (unter der Annahme, dass ihre Gewichte positiv sind).
- NURBS-Flächen sind differenzierbar.

Die Eigenschaften von T-Spline-Flächen folgen aus den Eigenschaften der NURBS-Flächen. Einige hiervon werden für den Anwendungsfall der Bathymetrien genauer betrachtet:

Lokale Verfeinerung Das vorgestellte Approximationsschema erzeugt progressiv verfeinerte T-Netze für Bathymetrien. Da die Änderungen lediglich einen lokalen Effekt haben dürfen, ist die lokale Verfeinerung von Flächen eine wesentliche Voraussetzung.

Affine Invarianz Eine Fläche wird durch die geometrische Transformation ihres Kontrollpunktnetzes beeinflusst. In Abschnitt 3.3.2 wird die nicht-affine, lokale Warping-Transformation eingeführt, die zwingend invertierbar sein muss, da sowohl eine Vorwärts- als auch eine Rückwärtstransformation durchgeführt werden. Beide Eigenschaften garantieren die Funktion der Warping-Operation.

Eigenschaft der konvexen Hülle Eine T-Spline-Fläche liegt stets in der konvexen Hülle des modellierenden T-Netzes. Das T-Netz wird daher flächendeckend über dem Gebiet der Bathymetrie erzeugt.

2.2.7 Klassifikation von T-Spline-Flächen

Sederberg et al. und Aizeng [3] [77] klassifizieren T-Spline-Flächen nach dem im Folgenden beschriebenen Schema.

Standard-T-Spline-Flächen

Für eine Standard-T-Spline-Fläche gilt $\forall i : w_i = 1$ und $\sum_{i=0}^{n-1} w_i \cdot N^K[\sigma_i, \tau_i](s,t) = 1$ (Partition der Eins) für alle Kontrollpunkte i. Aus Funktion (2.7) folgt somit für Standard-T-Spline-Flächen

$$\mathbf{t}_{\mathrm{st}}(s,t) = \sum_{i=0}^{n-1} \mathbf{p_i} \cdot N^K[\sigma_i, \tau_i](s,t). \tag{2.9}$$

Diese können somit als nicht-rationale (B-)Spline-Flächen mit lokalen Knotenwerten verstanden werden.

Semi-Standard-T-Spline-Flächen

Für Semi-Standard-T-Spline-Flächen gilt: $\exists w_i \neq 1$ mit $\forall i : w_i \geq 0$
und $\sum_{i=0}^{n-1} w_i \cdot N^K[\sigma_i, \tau_i](s,t) = 1$. Dies führt zu

$$\mathbf{t}_{\mathrm{sst}}(s,t) = \sum_{i=0}^{n-1} \mathbf{p_i} \cdot w_i \cdot N^K[\sigma_i, \tau_i](s,t). \tag{2.10}$$

Nicht-Standard-T-Spline-Flächen

Für Nicht-Standard-T-Spline-Flächen gilt allgemein lediglich

$$\sum_{i=0}^{n-1} w_i \cdot N^K[\sigma_i, \tau_i](s,t) \neq 0. \tag{2.11}$$

Die Definition der Nicht-Standard-T-Spline-Fläche entspricht somit der vorab eingeführten Definition der T-Spline-Fläche (2.7).

Diese Klassen spezialisieren die T-Spline-Flächen vollständig. Die Trennung zwischen Standard- und Nicht-Standard-T-Spline-Flächen ist innerhalb der Flächenrückführung für die Durchführung der linearen Optimierung wichtig (Abschnitt 4.3). Die Berücksichtigung von Unstetigkeiten am Rand der T-Spline-Fläche, bedingt durch wiederholte Knotenwerte, erfordert hierbei die Verwendung von Nicht-Standard-T-Spline-Flächen.

2.2.8 Operationen auf NURBS- und T-Spline-Flächen

An dieser Stelle wird ein Blick auf ausgewählte Anwendungen und Operationen für NURBS- und T-Spline-Flächen geworfen. Der Fokus liegt auf Anwendungen, die für die Generierung von T-Spline-Flächen für Bathymetrien notwendig sind.

Verfeinerung von Flächen

Die Verfeinerung einer NURBS-Fläche ist ein wesentlicher Schritt für die Erfassung von lokalen Details einer untersuchten Bathymetrie. Sie beruht auf dem Einfügen von Kontrollpunkten (*knot insertion*) in einem bestehenden Kontrollpunktgitter für NURBS-Flächen [26] [66]. Der Ablauf gleicht i. W. dem Einfügen von Kontrollpunkten bei Kurven, so dass die Unterteilung einer Fläche letztlich dem Einfügen einer Schar von Reihen- bzw. Spaltenpolygonen im Kontrollpunktgitter entlang einer Parameterrichtung der Fläche entspricht.

Eine Anwendung der lokalen Verfeinerung bei T-Spline-Flächen ohne globale Änderungen an der Reihen-Spalten-Struktur eines Kontrollpunktgitters beschreiben Sederberg et al. in [73] [75].

Reduktion von Kontrollpunkten

Der umgekehrte Weg der Verfeinerung, d. h. das Entfernen von Kontrollpunkten aus dem Kontrollpunktnetz (*knot removal*), ist ein ebenso üblicher Vorgang [23] [81] [59]. Auf diese Weise können Top-Down-Ansätze der Approximation realisiert werden. Hierbei wird ein hochaufgelöstes Kontrollpunktnetz sukzessiv lokal vergröbert, solange ein Qualitätskriterium für die Approximation ausreichend erfüllt ist. Vergleichbare Schritte bei der Vergröberung von T-Spline-Flächen beschreiben [15] [75] [83].

Verbindung

Ausgehend von zwei beliebigen Flächen für Objekte soll eine gemeinsame Fläche konstruiert werden, die einen anwendungsspezifischen Übergang zwischen diesen Objekten beschreibt. Dies kann beispielsweise das Aneinanderfügen von Teilstücken einer Bathymetrie oder von beliebigen Gebilden in der mechanischen oder medizinischen Modellierung sein. Die „Verbindung“ (z. T. auch „Überblendung“, „Verblendung“, *blending*, *merging*) beschreibt hierfür Verfahren zum „[...] Einsetzen von Flächenstücken in der Umgebung der Schnittkurve mit dem Ziel eines glatten Übergangs“ [1]. Diese Schnittkurve kann real existieren (bei sich überschneidenden Geometrien) oder sie entspricht dem Schnitt der imaginär bis zu einem gedachten Schnittbereich erweiterten Flächen. Eine Verbindung von Flächen erzeugt oftmals eine Geometrie mit zusätzlichen Kontrollpunkten im Schnittbereich. Sederberg et al. beschreiben die Verbindung von NURBS-Flächen zu einer kontinuierlichen, wasserdichten (*water tight*) und stetigen T-Spline-Fläche [76] [42].

2.2.9 Forschungsbedarf

Vielzählige Forschungsgebiete betrachten die computergestützte Modellierung von Spline-Flächen für dreidimensionale, geographische Geländemodelle. Allerdings vermissen die gängigen Modellierungen mit NURBS- oder Bézier-Flächen oftmals die Fähigkeit, lokale Details effektiv herauszuarbeiten. Stattdessen wird lediglich eine globale Verfeinerung durch das Einfügen von Reihen- oder Spaltenpolygonen im

Kontrollpunktgitter betrachtet. T-Spline-Flächen hingegen unterstützen die lokale Verfeinerung im Kontrollpunkt-T-Netz direkt.

Aus dem Anwendungsfall der Modellierung und Reduktion von bathymetrischen Geländemodellen ergeben sich darüber hinaus Probleme in der Skalierbarkeit der angebundenen Rechenmodelle (z. B. für die Simulation von Strömungsdynamik oder Stofftransport) bezogen auf ihre Darstellung (im Vergleich zu NURBS-Flächen und den dort verwendeten Kontrollpunktgittern) als auch in den dafür notwendigen Rechenressourcen. Diese Aspekte treten besonders bei sehr großen und inhomogen verteilten Datenmengen auf. Fortgeschrittene Ansätze für die Verarbeitung großer Datenmengen, wie sie aus (hydrographischen) Vermessungen oftmals entstehen, basierend auf der Modellierung mit T-Spline-Flächen, sind nicht bekannt und sind daher Kern dieser Arbeit.

2.3 Ein Approximationsschema für die Flächenrückführung

Das Ziel jeder allgemeinen Flächenrückführung ist die Modellierung einer geometrischen Freiformfläche, die eine vorhandene Referenzgeometrie wie eine Batyhmetrie (annähernd) gut erfasst. Diese Geometrie kann in Form beliebiger parametrischer Flächen wie Spline-Flächen oder Rasterdaten mit bivariater Interpolation vorliegen. Sehr oft basiert die Approximation auf einer Punktwolke, die z. B. aus dem dreidimensionalen Scan eines Objekts entsteht. Bathymetrische Geländedaten, insbesondere Primärdaten wie Sonarpeilungen oder Laserscans, liegen somit als flächenhafte Punktdaten vor.

Ein Approximationsschema für die allgemeine Flächenrückführung beruht auf der Wiederholung von zwei abwechselnden Phasen bis zum Erreichen der erwarteten Approximationsgüte:

- Verfeinerung einer geometrischen Fläche bzw. ihres Kontrollpunktnetzes, und
- Optimierung der erzeugten Geometrie anhand eines Gütekriteriums.

Beide Schritte sind modular. In dieser Arbeit wird ein Approximationsschema für die Flächenrückführung mit T-Spline-Flächen beschrieben (Abschnitt 4.1). Darauf aufbauend folgen die Ansätze zur Optimierung der erzeugten Geometrie (Abschnitt 4.3).

2.3.1 Approximation mit rationalen Spline-Flächen

Piegl und Tiller beschreiben die Approximation durch NURBS-Flächen als einen iterativen Vorgang der Anpassung eines Kontrollpunktnetzes [66]. Letztlich wird eine interpolierte NURBS-Fläche an eine Menge an Stützpunkten angenähert.

Die Autoren beschreiben einen Bottom-Up-Ansatz für ein globales Approximationsschema. Gegeben ist eine Referenzfläche durch eine Anzahl an Messpunkten (*samples*). Eine globale Approximation von „unten“ (minimales Netz) nach „oben“ (optimales Netz) erfordert die folgenden Schritte:

1. Initiiere ein Kontrollpunktnetz mit minimaler Anzahl an Kontrollpunkten.
2. Optimiere das Kontrollpunktnetz bezogen auf den zu minimierenden Fehler der Differenz zwischen der NURBS-Fläche und den Messpunkten.
3. Bestimme den Fehler der Differenz zwischen NURBS-Fläche und Messpunkten.

4. Falls der berechnete Differenzfehler einen Schwellwert unterschreitet, dann beende die Approximation. Sonst verfeinere das Kontrollpunktnetz und gehe zurück zu Schritt 2.

Die Methode der Verfeinerung wird von den Autoren nicht näher beschrieben, da diese modular und somit wählbar ist. Aus Effizienzgründen wird in dieser Arbeit stets ein Bottom-Up-Ansatz verwendet. Andernfalls müssen bereits zu Beginn hochaufgelöste T-Netze konstruiert werden, die mit höherem Rechenaufwand verbunden sind.

Das vorgestellte Approximationsschema für T-Spline-Flächen weicht von den beschriebenen Ansätzen ab, da keine globalen Netzoperationen wie das Einfügen von Reihen- oder Spaltenpolygonen in Kontrollpunktgittern stattfinden. T-Spline-Flächen ermöglichen insbesondere nur lokale Verfeinerungen. Ein zusätzlicher Freiheitsgrad in der Modellierung mit T-Spline-Flächen ist die Modifikation der Gewichte der Kontrollpunkte (siehe Abschnitt 4.5.1). Eine Anpassung des Ansatzgrads ist allerdings nicht vorgesehen.

2.3.2 Fehlermaße

Ein Maß für die Qualität einer erzeugten T-Spine-Fläche stellt die Abweichung zur Bathymetrie dar. Die hier verwendete Metrik bestimmt den Abstandsfehler zwischen zwei Geometrien, d. h. einer erzeugten T-Spline-Fläche und einer kontinuierlichen Bathymetrie. Alternative Fehlermaße, d. h. solche die nicht auf Distanzfehlern beruhen, verwenden z. B. geometrische Aspekte wie extreme Krümmungsradien [52] oder die Varianz der Tiefenwerte [67].

Diskreter Fehler

Sei $\mathbf{X} = \mathbf{f_1}(s,t) - \mathbf{f_2}(s,t)$ ein Vektor zwischen zwei Flächenpunkten der C^0-stetigen Flächen $\mathbf{f_1}$ und $\mathbf{f_2}$ an einem gemeinsamen Parameterpaar (s,t) .

Der globale Fehler ergibt sich über eine Menge an gemeinsamen Parameterpaaren $S = \{(s_i,t_i)\} \subseteq \mathbb{R}^2$ der beiden Flächen. Die berechneten diskreten Flächenpunkte sind somit $\mathbf{f_1}(s_i,t_i)$ bzw. $\mathbf{f_2}(s_i,t_i)$ und der punktuelle Fehler zwischen diesen ist

$$e(s_i,t_i) := d(\mathbf{f_1}(s_i,t_i) - \mathbf{f_2}(s_i,t_i)). \tag{2.12}$$

Der punktuelle Fehler e ergibt sich hierbei über eine Funktion $d(\mathbf{X}) \to \mathbb{R}$ als Projektion des Differenzvektors auf einen skalaren Wert. Üblich ist $d(\mathbf{X}) := ||\mathbf{X}||$, d. h. die Berechnung einer beliebigen Vektornorm (s. u.). Für die alleinige Einschränkung auf die Fehler der z-Werte gilt $d(\mathbf{X}) := |z(\mathbf{f_1}(s,t)) - z(\mathbf{f_2}(s,t))|$, also der Betrag der Tiefenwertdifferenzen.

Über alle diskretisierten Flächenpunkte ergibt dies den globalen Fehlervektor $\mathbf{E} := (e_1 \dots e_{|S|})^T$. Die Komponenten von $\mathbf{E}$ sind somit die einzelnen (projizierten) Differenzfehler an jedem gemeinsamen Parameterpaar. Zwei Flächen sind somit identisch, genau dann wenn $\mathbf{E} = \mathbf{0}$ ist.

Anhand einer ebenfalls frei wählbaren Norm für den globalen Fehlervektor ergibt sich das Maß für die Qualität der Approximation. Ziel ist es, den globalen Fehler zu minimieren.

Normen

Die allgemeine p-Norm lautet

$$||\mathbf{E}||_p := \left(\sum_{i=1}^{|S|} |e(s_i, t_i)|^p \right)^{\frac{1}{p}}. \tag{2.13}$$

Für $p = 1$ ist die L1-Norm als Manhattan- oder Taxi-Norm bekannt. Ihr Vorteil liegt in der effizienten Berechnung. Die L2-Norm (d. h. $p = 2$) entspricht der euklidischen Länge des Vektors, oft reicht für die Effizienz bereits die quadrierte Form $||\mathbf{E}||_2^2$.
Die Maximumnorm gibt die Komponente des Fehlervektors mit größtem Betrag an:

$$||\mathbf{E}||_{\max} := \max_{i=1...|S|} |e_i|. \tag{2.14}$$

Dies entspricht dem gemeinsamen Flächenpunkt mit größter Abweichung. Eine Optimierung nach der Maximumnorm minimiert somit die maximale Abweichung zwischen der Bathymetrie und der T-Spline-Fläche.

2.3.3 Flächenrückführung

Allgemein bezeichnet „Flächenrückführung“ den Prozess, eine unbekannte geometrische Oberfläche (d. h. die umfassenden (Teil-)Flächen eines Objekts) durch ein mathematisches Modell bestmöglich anzunähern. Die Geometrie ist durch eine endliche Menge an verteilten Punkten im dreidimensionalen Raum $\mathbf{x} \in \mathbb{E}^3$ gegeben. Diese Punkte stammen heutzutage oft aus der räumlichen Abtastung eines Objekts mit sensorischen Mitteln wie Laser, Sonar oder der Photogrammetrie. Lodha und Franke [58] unterscheiden die Abbildungen hierbei zwischen

- der Flächenrückführung $\mathbf{f}(s, t) \rightarrow (x\, y\, z)^T \in \mathbb{E}^3$ (*surface reconstruction*), die jeden gemessenen zweidimensionalen Punkt auf einen dreidimensionalen Punkt einer Oberfläche abbildet,
- der Funktionsrückführung $f(s, t) \rightarrow x \in \mathbb{R}$ (*function reconstruction*), die jedem gemessenen zweidimensionalen Punkt einen (Höhen-)Wert zuweist, und
- der Fläche-auf-Fläche-Rückführung $\mathbf{f}(s, t, u) \rightarrow (x\, y\, z)^T \in \mathbb{E}^3$ (*surface on surface problem*), die jeden gemessenen dreidimensionalen Punkt auf einen Punkt einer Oberfläche abbildet.

Da die betrachteten Bathymetrien keine Überhänge besitzen, existiert an jedem Messpunkt höchstens ein Tiefenwert. Nicht zuletzt führt die Tatsache, dass die Vermessungen oftmals entlang von Streckenprofilen durchgeführt werden, zu Lücken innerhalb der Vermessungsgebiete. Genau genommen sind die zu betrachtenden Flächenrückführungen somit Funktionsrückführungen, die jedem Punkt der modellierten Flächen einen Tiefenwert zuweisen[5].
Die Autoren führen folgende Modellansätze für die Funktionsrückführung auf [58]:

- Polynomielle bzw. parametrische Spline-Flächen
- Algebraische Flächen

[5] In dieser Arbeit soll der Begriff der Flächenrückführung somit gleichbedeutend sein, um die Anwendung von T-Spline-Flächen sprachlich einzuordnen.

- Radiale Basisfunktionen
- Methoden der Distanzgewichtung (*Shepard*, *Inverse Shepard*)
- Unterteilungsflächen (z. B. *Catmull-Clark Subdivision Surfaces*).

Beispielhafte Arbeiten für die oben aufgeführten Ansätze finden sich in [99] [46] [63]. T-Spline-Flächen verfügen als parametrische Flächen über zusätzliche Eigenschaften der Unterteilungsflächen (in Bezug auf die Verfeinerung von Elementen) und sind daher Kern der vorgestellten Flächenrückführung.

Flächenrückführung mit NURBS-Flächen

Sehr gute, weiterhin aktuelle Übersichten der Flächenrückführung auf Basis von NURBS-Flächen finden sich in [85] [38] [18] [29]. Veröffentlichungen, die besonders die Flächenrückführung über die Methode der kleinsten Quadrate und die Einbindung eines zusätzlichen Fairness-Terms betrachten, sind [11] (Fokus auf Geländemodellen), [16] (Lösung mittels der Methode der nichtlinearen kleinsten Quadrate) oder [24] (B-Spline-Fläche über triangulierte Punktwolken).

Flächenrückführung mit T-Spline-Flächen

Weitere Autoren haben sich speziell mit der Flächenrückführung mit T-Spline-Flächen befasst. Ein großer Teil der Arbeiten verwendet die Methode der kleinsten Quadrate, um vorhandene T-Netze an die Referenzgeometrie anzupassen. Sederberg et al. haben bereits in einem frühen Beitrag vorgeführt, wie die schrittweise Vergröberung von T-Netzen in Verbindung mit einer Flächenrückführung durchgeführt wird [75]. Spätere Arbeiten greifen dies auf [82] [22] [36] [54] [98]. Wang et al. [84] verwenden im Vergleich zu dieser Arbeit einen prinzipiell ähnlichen Approximationsansatz. Als Datenbasis verwenden die Autoren allerdings Dreiecksnetze, und die Positionen der Verfeinerung werden durch die lokale Krümmung vorgegeben. T-Spline-Flächen aus Punktwolken erzeugen [90] [89].

Die Flächenrückführung von T-Spline-Flächen weicht insofern von NURBS-Flächen ab, als dass neben der Lage der Kontrollpunkte und deren Gewichte die lokalen Knotenwerte wesentliche Faktoren darstellen. Die Struktur der T-Netze (z. B. lokale Dichte, Ausdehnung, ...) ist darüber hinaus gekoppelt an die Verteilung der Knotenwerte, d. h. Änderungen am T-Netz erfordern aktualisierte Knotenwerte, und umgekehrt erzwingen neu eingefügte Knotenwerte topologische Änderungen im T-Netz. Diese stellen ebenfalls ein prinzipielles Optimierungskriterium dar. Grundsätzliche Probleme hieraus werden in Abschnitt 5.3 in Bezug auf hängende Knoten und einer angepassten Herangehensweise thematisiert.

2.3.4 Anwendung und Forschungsbedarf

Die Flächenrückführung ist ein allgemein gut untersuchtes Thema in der geometrischen Modellierung. In den überwiegenden Arbeiten finden iterative Verfahren Anwendung, die eine zu optimierende Fläche (d. h. ihr zugrundeliegendes Kontrollpunktnetz) an eine gegebene Referenzgeometrie sukzessiv anpassen. Die eingesetzten Approximationsschemata umklammern dabei die spezifischen Optimierungsverfahren. Ein methodisch einfaches Verfahren ist die Berechnung einer Lösung anhand der Methode der kleinsten Quadrate, die wohl untersucht und in diversen numerischen Programmen und Softwarebibliotheken generisch zur Verfügung steht. Die hierfür notwendige Rechenleistung

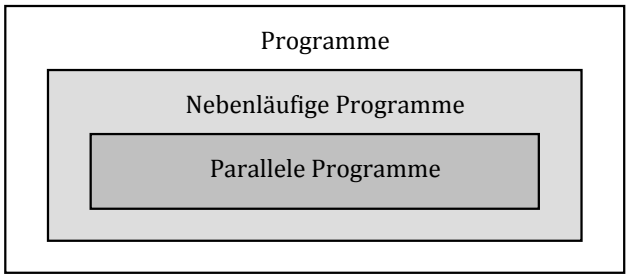


Abbildung 2.7: Modelle paralleler Programme nach [30]

steigt jedoch überlinear mit der zu optimierenden Netzgröße[6]. Hieraus ergibt sich ein Bedarf nach der effizienten Umsetzung dieser Methode und einer möglichen Benennung von technischen Schranken.

Für NURBS-Flächen stehen unterschiedliche Verfahren der Flächenrückführung zur Verfügung, allerdings umgehen diese oftmals nicht die in Abschnitt 2.2.9 aufgeführten Probleme der lokalen Detailausarbeitung und der Skalierbarkeit. Auf T-Spline-Flächen basierende Flächenrückführungen sind bereits in einzelnen Forschungsarbeiten zu finden. Diese beruhen jedoch nur auf geringen Datengrößen und konzentrieren sich dabei auf andere Themen abseits der bathymetrischen Modellierung, wie z. B. die Bildverarbeitung. Hier sind weitere Betrachtungen hinsichtlich des Anwendungsfalls notwendig.

In dieser Arbeit werden daher die Flächenrückführung mit linearer Optimierung und das sogenannte Dragging-Verfahren zum progressiven Verschieben von Kontrollpunkten (bzw. ihrer Tiefenwerte) bei T-Spline-Flächen analysiert. Nicht zuletzt werden die numerischen Aspekte dieser Methoden betrachtet, die aus der Modellierung von großräumigen Bathymetrien heraus entstehen.

2.4 Beschleunigung durch massiv-parallele Programme

Parallele Anwendungen beruhen sowohl auf parallelisierbaren Programmen, als auch auf einer Hardware, die Parallelität direkt unterstützt. Es folgt ein Blick auf die Umsetzung mittels GPGPU-Architekturen und speziell die grundlegenden Konzepte der OpenCL-Schnittstelle.

2.4.1 Parallele Programme

Gaster et al. [30] beschreiben drei in sich verschachtelte Programmmodelle (Abbildung 2.7). Die Menge aller Programme umfasst sämtliche ein- und nebenläufigen Programme. Ein nicht-paralleles Programm führt eine Reihe von Anweisungen seriell durch. Ein (algorithmisches) Problem wird hierbei ggf. in einzelne Unterprobleme (z. B. *divide and conquer*) oder Arbeitsschritte (*pipelining*, *streams*) zerlegt und diese werden in einer deterministischen Reihenfolge auf den verfügbaren Rechenknoten verarbeitet.

[6] Im Kern beruht die Methode der kleinsten Quadrate auf der effizienten Berechnung der Zerlegung einer Optimierungsmatrix.

Allgemein wird zwischen der Nebenläufigkeit (*concurrency*) und der Parallelität von Programmen unterschieden:

- Nebenläufige Programme basieren auf grundlegenden Prinzipien der Lastverteilung, der Synchronisation und dem Nachrichtenaustausch auf und zwischen den Rechenknoten (s. u.).
- Parallelprogramme zielen auf eine höchstmögliche Ausnutzung aller Hardware-Ressourcen. Operationen werden auf unterschiedliche Rechenknoten ausgelagert und somit gleichzeitig verarbeitet.

Dabei gilt durchgängig die Beziehung der Inklusion: ein Parallelprogramm ist immer ein nebenläufiges Programm, da ein Verteilen von Daten und das Sammeln von Ergebnissen Kommunikation zwischen den Rechenknoten erfordert. Matrixmultiplikationen oder die Fast-Fourier-Transformation sind Beispiele für Algorithmen mit theoretisch bestmöglicher Parallelität, da die einzelnen Rechenschritte unabhängig voneinander auf isolierten Kernen erfolgen können [30]. Praktische Umsetzungen erzielen diese Parallelität allerdings selten. Nebenläufigkeit ist nicht unmittelbar mit Parallelisierung gleichzusetzen. So können beispielsweise Verzögerungen in der Netzwerkkommunikation oder allgemeine Synchronisierungsprobleme große Lücken in der zeitlichen Abfolge erwirken und folglich nachhaltig die Parallelverarbeitung stören.

Ein Knoten entspricht einer Recheneinheit. Dies können isolierte (Multi-)Kerne (*core*) in einem Prozessor (CPU / GPU), Großrechner / Mainframes, lokale Cluster-Systeme oder weltweit verteilte Grid-Architekturen sein. Die wesentlichen Probleme der Nebenläufigkeit bleiben erhalten, skalieren allerdings erheblich mit der Größe der Systeme. Cache-Inkohärenzen[7] erzwingen einen Datenaustausch, und Netzwerk- bzw. Speicheroperationen führen zu Zugriffsblockaden [51]. Nebenläufige Programme sind weiterhin auf Single-Core-CPUs ausführbar. Parallelität wird hierbei i. A. durch zeitliche Fenster simuliert, die jedem Knoten eine bestimmte Zeit zur Verfügung stehen.

Ein paralleler Algorithmus ist prinzipiell ein serieller Algorithmus, dessen Schritte auf verschiedene Recheneinheiten verteilt werden. Diese Verteilung kann horizontal über die Eingabedaten bzw. die erwarteten Ausgabedaten erfolgen (Beispiele wären Flächenberechnung, *Raytracing*, *Map-Reduce*, ...), oder vertikal über die Schritte des Algorithmus (Aufbau einer Prozess-*Pipeline*, *Load Balancing*, ...). Algorithmen sind zumeist dann effektiv parallelisierbar, wenn sie die Aspekte der Nebenläufigkeit (z. B. Knoten-übergreifende Synchronisation) nur minimal tangieren. Eine gute Verteilung und kleine, lokal[8] verwendete Datenblöcke bewirken ebenfalls eine gute Effizienz.

2.4.2 Vektorrechner

Ein Vektorrechner ist ein (zunächst abstraktes) Rechenwerk, das simultane Operationen auf einem Datenvektor durchführt. Unter einem „Vektor“ im hier verwendeten Sinn versteht man einen schematisch identischen Datensatz mit einer variablen Anzahl an Komponenten. Operationen werden stets auf dem gesamten Vektor komponentenweise durchgeführt, vergleichbar mit der mathematischen Vektoraddition als separate Addition aller Komponenten. Hauptsächlich finden Vektorrechner — gemäß der SIMD[9]-Direktive — ihren Einsatz im Ausführen von schnellen Befehlsketten auf

[7] Zeitintensives Auslagern und Füllen von gemeinsam genutzten Speicherblöcken
[8] Gemeint ist die zeitliche und räumliche Lokalität bei Computerprogrammen.
[9] *Single Instruction - Multiple Data*

großen Datenmengen. Alltägliche Beispiele sind u. a. in der Signalverarbeitung (z. B. Audio / Video), in der Netzwerktechnik (Routing) oder in der Kryptografie zu finden.

2.4.3 GPGPU-Programmierung

Grafikkarten stellen prinzipiell einen vollwertigen Rechner dar, da sie über eine GPU (*graphics processing unit*) als eigenständigen Prozessor, einen dedizierten Speicher und die notwendige I/O-Peripherie (*input / output*) verfügen. Ursprünglich gedacht als Beschleuniger für die Bilderzeugung[10], sind sie heutzutage vielseitig einsetzbare und günstige Vektorrechner.

Frühzeitige Versuche, die Rechenleistung von GPUs für wissenschaftliche Berechnungen einzusetzen, scheiterten z. T. an der mangelnden Unterstützung für generische Programmierung abseits der Geometrieverarbeitung. Shader-Programmiersprachen wie *C for Graphics* (*Cg*) und *High Level Shading Language* (*HLSL*) bieten nur rudimentäre Programmiermodelle, deren Zweck letztlich die Berechnung von Vertex- und Pixeloperationen ist. Angetrieben von den Herstellen NVidia Corporation und ATI Technologies[11] werden Schnittstellen definiert, die vielseitigere Berechnungen mit GPUs zulassen. Diese werden als *General Purpose Computation on Graphics Processing Unit* (GPGPU) bezeichnet. Die beiden populärsten GPGPU-Schnittstellen sind *Compute Unified Device Architecture* (*CUDA*), beschränkt auf NVidia-Hardware [19], und *Open Computing Language* (*OpenCL*), die von der herstellerübergreifenden Khronos Group betreut wird [33].

Nach dem heutigem Stand der Entwicklung besitzen Grafikkarten deutlich effektivere Komponenten für wissenschaftliche Berechnungen als frühere Generationen. Aktuelle Entwicklungen bringen Hochleistungs-GPUs für den wissenschaftlichen Einsatz[12] und hochgradig skalierbare Architekturen hervor, die mehrere GPUs in einem System integrieren (z. B. NVidia Tesla). In Verbindung mit schnellen Speicherbausteinen überragen die erzielten Rechenleistungen vergleichbar teure CPU-Systeme um den Faktor mehrerer Tausend und dringen in Bereiche vor, die bis vor wenigen Jahren noch Supercomputern vorbehalten blieben. Tabelle 2.2 stellt die „historische“ Entwicklung der GPGPU-Entwicklung dar [72].

In Kapitel 6 wird OpenCL als Schnittstelle für massiv-parallele Teilaufgaben in die Reduktion bathymetrischer Flächen integriert. Weitere Nutzung von GPGPU-Beschleunigung, speziell in der Geländemodellierung, finden sich in [91] [68] [41] [50]. Eine Gruppe um Zhang befasst sich mit der binären Kodierung und Kompression von Geländedaten (speziell Rasterdaten) mittels GPGPU-basierter Implementierungen [93] [96] [95] [92].

2.4.4 OpenCL

Die *Open Computing Language* ist eine GPGPU-Schnittstelle mit dem Ziel, vorhandene Rechenkapazität durch ein einheitliches Modell zur Verfügung zu stellen [72] [64] [30] [80] [33]. Sie bietet eine virtuelle Architektur, die letztlich auf konkrete Prozessoren und Speicherhierarchien abgebildet wird.

[10] 3D-Geometrie- und Videoverarbeitung und die Ausgabe auf Monitor(en)
[11] Heute Teil von Advanced Micro Devices (AMD)
[12] Der Schwerpunkt liegt auf der Gleitkommaleistung.

Zeitraum	Ereignis
2001	IBM Power4 (Erster „echter“ Multikernprozessor)
2005	Intel Pentium D (Erster Desktop-Doppelkernprozessor)
2005 – 2008	OpenCL Working Group (Apple, AMD, IBM, Intel, NVidia)
Juni 2007	CUDA 1.0 SDK
Dezember 2008	OpenCL 1.0 Spezifikation
April 2009	NVidia OpenCL SDK 1.0
August 2009	ATI / AMD OpenCL SDK 1.0 Unterstützung in Max OS X 10.6
2010	Tianhe 1A (Erster Supercomputer mit GPGPU-Unterstützung)
November 2011	OpenCL 1.2 Spezifikation
Juni 2014	Tianhe 2 (Platz 1 der HPC Top 500, ebenfalls GPGPU-basiert)
November 2013	OpenCL 2.0 Spezifikation
März 2015	OpenCL 2.1 Spezifikation

Tabelle 2.2: Zeitlinie GPGPU-Entwicklung

Scarpino beschreibt die wesentlichen Vorteile [72]:

Portabilität Die Effektivität von OpenCL liegt in der Portierung der Schnittstelle auf eine Vielzahl an Architekturen wie Grafikbeschleunigern und Server- / Desktop-Prozessoren und vereinzelten Signalprozessoren. Das Paradigma „*Write once, run on anything*“ steht für die Vielfältigkeit der unterstützten Plattformen. Einmal erstellter Programmcode ist auf allen verfügbaren Systemen einsetzbar (ein Vorteil gegenüber CUDA).

Vektor-Datentypen OpenCL erweitert den Sprachumfang von ISO C99 um Datentypen und Befehle zur vektorisierten Datenverarbeitung. Eine effektive Parallelverarbeitung findet bereits auf Ebene atomarer Operationen statt (z. B. parallele Addition von Vektorkomponenten).

Parallelität Programme unterstützen Daten- und Task-Parallelität. Unterschiedliche und unabhängige Operationen werden simultan auf getrennten Daten durchgeführt. Das *Execution Model* sichert die isolierte Ausführung der Kernel.

Das vollständige OpenCL-Framework besteht aus einzelnen Komponenten [64]:

- Die **OpenCL Plattform-API**[13] konfiguriert das Gastsystem und initialisiert den Kontext.
- Die **OpenCL Runtime-API** verwaltet den Kontext und die Kernel-Aufrufe zur Laufzeit.
- **OpenCL C** ist die erweiterte Programmiersprache ISO C99.

[13] Application Programming Interface

Vielfältige Anwendungen greifen bereits auf GPGPU-Beschleunigung zurück, und bevorzugen dafür OpenCL aus den oben genannten Gründen der Portabilität und Vereinheitlichung. Häufig finden sich OpenCL-basierte Anwendungen in der Wissenschaft und Technik. Weitere Beispiele aus dem Heimbereich sind Bild-, Audio- und Videoverarbeitung, da diese ebenfalls auf großen Datenmengen vektorisiert arbeiten. Kapitel 6 geht auf die relevanten Bestandteile der virtuellen Architektur von OpenCL genauer ein. Die Reduktion von Bathymetrien verwendet parallele Berechnungen, die auf die OpenCL-Architektur transferierbar sind.

2.4.5 Forschungsbedarf

Die Umsetzung von beschleunigten Berechnungen auf Grafikprozessoren erfordert ein hohes Maß an Anpassungen des entwickelten bzw. zu transformierenden Programmcodes. Neben der eigentlichen Aufgabenstellung, und den dazu benötigten Algorithmen und Datenstrukturen, ist oftmals eine Plattform-gerichtete Entwicklung nötig. OpenCL stellt hierzu lediglich einen allgemeinen Rahmen bereit. Standardisierte Bibliotheken für spezielle Aufgaben existieren kaum, der Fokus liegt hier oftmals auf Algebra-Bibliotheken. Teile der vorliegenden Arbeit befassen sich dabei mit der Entwicklung von Algorithmen und Datenstrukturen für die Modellierung und Flächenrückführung mit T-Spline-Flächen. Hierbei wird auf eine Hardware-unabhängige und somit übertragbare Lösung geachtet.

2.5 Motivation dieser Arbeit

Zusammengefasst ergeben sich aus den vorgestellten Teilgebieten einzelne Aspekte, die für eine Modellierung mit T-Spline-Flächen sprechen.

Stetige Flächen sind in vielzähligen Bereichen von Bedeutung. Neben der reinen Visualisierung steht die geometrische Stetigkeit, beispielsweise bei Freiformflächen im Fahrzeugentwurf, im Vordergrund, da sie sowohl ästhetische Aspekte als auch die zugrundeliegenden Umformverfahren in der Produktion berücksichtigen muss. Hydrografische Geländemodelle sind oftmals relevant in der Strömungssimulation (*computational fluid dynamics*). Sie ermöglichen die Simulation des Stofftransports (z. B. Sedimente oder Salzintrusion) entlang einer glatten Oberfläche [62] [101] [100].

T-Spline-Flächen sind eine Option für Freiformflächen mit wählbarer Stetigkeit. Als konkretes Beispiel zeigen Sederberg et al. die Konstruktion von Schiffsrümpfen speziell mit T-Spline-Flächen [74]. Nicht zuletzt ist die isogeometrische Analyse ein neuartiges Optimierungsverfahren, das die beidseitigen Vorteile der Modellierung mit finiten Elementen und die Geometrie der NURBS-Flächen aufgreift [40] [20].

Die vorgestellte Anwendung beruht auf sehr umfangreichen Primärdaten aus geo- oder hydrografischen Messreihen. Neben der reinen Masse weisen diese Daten in bestimmten Kontexten eine unregelmäßige Verteilung auf. Schiffsgebundene Messungen besitzen eine spurförmige Verteilung mit einer gebietsweise hohen Dichte an Messpunkten. Kombinierte Messungen, die sowohl schmale Fluss- als auch weiträumige Meeresgebiete erfassen, stellen ebenfalls eine Herausforderung dar.

Dies führt zu Modellen, deren praktische Integration durch den erhöhten numerischen Aufwand häufig nicht mehr zu rechtfertigen ist. Zorndt et al. beschreiben die Problematik der Datenreduktion, die sowohl aus der reinen Notwendigkeit, als auch aus der Situation der lokal dichteren Modellierung entsteht [101] [100]. Hieraus ergibt sich ein Bedarf an adaptiven Flächenmodellen, die lokale Gegebenheiten mit

der gleichen Sorgfalt wie großräumige Genauigkeit abdecken. Die freie Modellierung mit T-Spline-Flächen bringt hier ebenso einen Vorteil gegenüber Spline-Flächen mit starren Kontrollpunktnetzen. T-Netze basieren auf der adaptiven Zerlegung eines Parameterraums und ermöglichen somit die lokale Verfeinerung.

In den Voruntersuchungen hat sich gezeigt, dass bereits kleinere Modelle schnell an technische Grenzen stoßen. Dies umfasst diverse Aspekte der Modellierung und Berechenbarkeit. Fortschreitende lokale Verfeinerungen im T-Netz sind in der Modellierung gesondert zu berücksichtigen und erhöhen den Aufwand, im Vergleich mit einfachen Kontrollpunktgittern bei NURBS-Flächen, enorm. Teilberechnungen und Speicherbedarf skalieren überproportional zur Netzgröße. Wesentliche Arbeitsschritte greifen auf kompakte, allerdings in großem Umfang wiederkehrende Operationen zurück (z. B. Schnittberechnungen im T-Netz, Abschnitt 5.2), so dass hier die Notwendigkeit nach paralleler Unterstützung entsteht. Die Berücksichtigung einer realer Bathymetrie wird exemplarisch betrachtet, allerdings unterstützt diese bereits die Forderung nach der Parallelisierung des vorgstellten Prozesses durch OpenCL.

T-Spline-Flächen finden sich in zunehmend mehr Anwendungsgebieten. Die Verwendung in Digitalen Geländemodellen, speziell bei Bathymetrien mit variierenden Datendichten, bleibt allerdings weiterhin die Ausnahme[14]. Als Modell in Geoinformationssystemen finden T-Spline-Flächen keine Beachtung, obwohl dieses Werkzeug im technischen Entwurf bereits etabliert ist.

Die Reduktion von Geländedaten, basierend auf einer Filterung von Rohdaten oder einer Neuvernetzung, ist durchaus üblich. Teilweise werden hier geringe qualitative Verluste akzeptiert. Diese Arbeit greift auf eine hochaufgelöste Rastergeometrie zurück, im Bedarfsfall wird diese vorab durch Interpolation der Primärdaten erzeugt. Alternative Ansätze zur Kompression kommen aus der Bildverarbeitung und der Wavelet-Theorie [5] [12]. Der vorgestellte Prozess führt neuartige Ideen in das Feld der Reduktion von Digitalen Geländemodellen basierend auf Freiformflächen ein.

In eigenen Vorarbeiten wird die Nutzung von T-Spline-Flächen für bathymetrische Geländemodelle motiviert [7] [8]. Grundlegende Vorteile von T-Spline-Flächen sind die lokale Verfeinerung, die frei wählbare Topologie, die Konvertierung in gängige Spline-Flächen und die stetige Verbindung von NURBS-Flächen. Diese Aspekte schaffen die Möglichkeit einer umfassende Modellierung von bathymetrischen Geländemodellen unter Berücksichtigung lokaler Gegebenheiten. In [6] ist die Implementierung einer Datenstruktur basierend auf Halbkanten für T-Netze mit der CGAL-Bibliothek erläutert [25] [28]. Von Nachteil ist hierbei, dass die in CGAL verwendete Polyeder-Datenstruktur insgesamt zu generisch aufgestellt ist und sie im vorgestellten Anwendungsfall, bezogen auf ihre Effizienz, schlecht skaliert.

[14] Zheng et al. verweisen auf kein konkretes Anwendungsgebiet [98].

3 Reduktion bathymetrischer Geländemodelle

Das Ziel der Arbeit ist ein abgeschlossener Prozess für die Reduktion bathymetrischer Geländemodelle. Die Grundlage hierfür bildet ein zentrales Approximationsschema, das zu einer in Form von Rasterdaten gegebenen Referenzbathymetrie eine optimierte T-Spline-Fläche (bzw. deren Kontrollpunktnetz) mit wählbarer Güte erzeugt. Unterschieden wird zwischen einer direkten und einer indirekten Reduktion. Die direkte Reduktion umfasst die Schritte des einleitenden Imports, der optimierenden Flächenrückführung und der finalen Netz-Ausgabe. Umrahmt wird das in großen Teilen identische Vorgehen der indirekten Reduktion um Operationen aus der Bildverarbeitung für die Vor- und Nachbearbeitung der Bathymetrie (Abbildung 3.1). Die Verknüpfung von Methoden der Bild- und Geometrietransformation ist nach dem aktuellen Stand der Forschung kaum geläufig, speziell gilt dies im Anwendungsfall der bathymetrischen Modellierung.

3.1 Methoden der Bildverarbeitung

Teile der indirekten Reduktion verwenden Methoden der Computer-grafischen Verarbeitung von (Raster-)Bilddaten. Angepasste Methoden funktionieren daher vergleichbar für bathymetrische Rastergeometrien. Anstelle der diskreten Farbkanalinformationen besitzen diese allerdings Tiefenwerte aus einem kontinuierlichen Spektrum.

Im Folgenden werden die Methoden der Bildverarbeitung vorgestellt, die für diese Arbeit genutzt werden.

3.1.1 Warping

Wolberg definiert das *Warping* als eine Technik in der Bildverarbeitung, die räumliche (*spatial*) Transformationen auf einem Bild durchführt [86]. Hierzu werden zunächst markante Merkmalspunkte (*features*) in Bildern bestimmt, und diese mittels einer Warp-Funktion transformiert. Eine derartige Funktion erzeugt Verschiebungsvektoren, die jeden Punkt des Rasterbilds auf eine neue Koordinate abbilden. Beispiele für Warping finden sich in der Korrektur von Verzerrungen bei Kameralinsen, der medizinischen Bildverarbeitung (z. B. in dem Ausrichten von Tomografieaufnahmen) oder beim *Morphing* in der Videoverarbeitung. Gute Übersichten der verschiedenen Ansätze sind [87] [31] [69].

Natürliche Gewässerböden besitzen zumeist keine markanten, punktuell bestimmbaren Merkmale wie Kanten oder Ecken, und daher sind automatisierte Verfahren der Merkmalsextraktion schwierig durchzuführen. Aus diesem Grund wird ein Dreiecksnetz erzeugt, das über eingeführte Verschiebungspunkte aufgespannt wird. Diese Verschiebungspunkte geben den Ausgang der Verschiebung vor. Die Bestimmung der Verschiebungsvektoren an jedem Verschiebungspunkt folgt in Abschnitt 3.2.2. Zu jedem Rasterpunkt wird sein umfassendes Dreieck identifiziert und anschließend wird die kontinuierliche Verschiebung über die multivariate, baryzentrische Interpolation der Verschiebungsvektoren an den Dreieckspunkten gebildet.

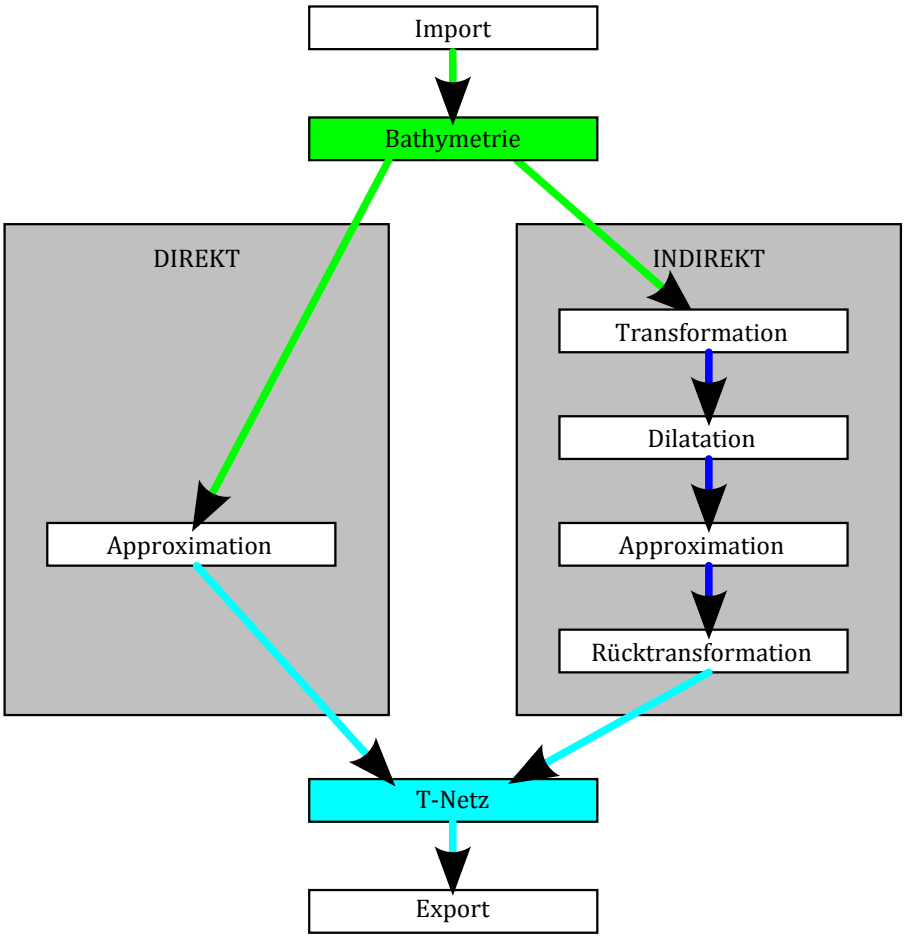


Abbildung 3.1: Darstellung der direkten und indirekten Reduktion

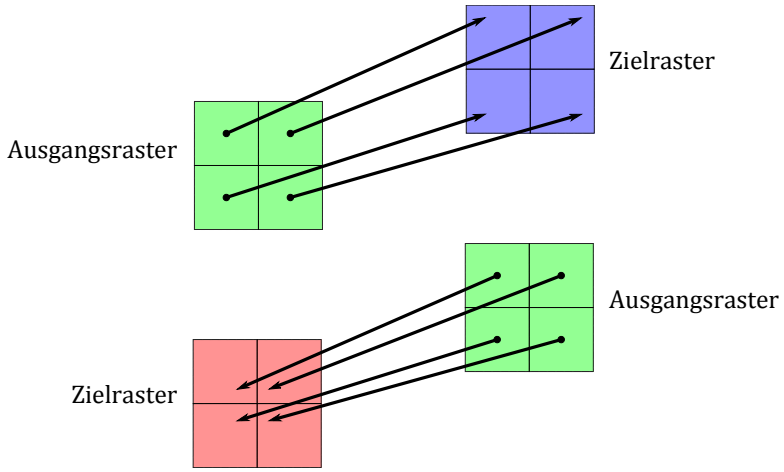


Abbildung 3.2: Vorwärts- (oben) und Rückwärtstransformation (unten) eines Ausgangsrasters (grün) auf ein Zielraster (blau bzw. rot) beim Warping

Die Transformation bewirkt nun die Abbildung der ursprünglichen Rasterdaten anhand des Verschiebungsfelds in eine neue Rastergeometrie. Ein markantes Problem stellt die Abbildung von ganzzahligen Punktkoordinaten auf (i. A.) nicht-ganzzahlige Punkte dar. Wolberg beschreibt die folgenden Ansätze für die Transformation [86]:

- Vorwärtstransformation (*forward mapping*): Jeder Punkt der Ursprungsgeometrie wird anhand des Verschiebungsfelds in die Zielgeometrie versetzt. In der Regel erfolgt die Abbildung auf nicht-ganzzahlige Rasterkoordinaten. Die Transformation wird durch Gauß-Splatting umgesetzt (Abschnitt 3.1.2).

- Rückwärtstransformation (*inverse mapping*): Jedem ganzzahligen Punkt der Zielgeometrie wird ein Punkt der Ursprungsgeometrie zugewiesen. Der Ansatz beruht auf der Invertierung des Verschiebungsfelds. Für jede ganzzahlige Rasterkoordinate in der Zielgeometrie wird ein (nicht-ganzzahliger) Punkt in der Urgeometrie bestimmt, der als Ausgang für die Transformation gilt.

Die Vorwärtstransformation verwendet die oben beschriebene Triangulation des Urgebiets für eine bivariate Dreiecksnetz-Interpolation der Verschiebungsvektoren (Abbildung 3.2). Die Rückwärtstransformation beruht auf der Triangulation des Zielgebiets und der Invertierung des Verschiebungsfelds.

Abbildung 3.3 stellt das Problem der nicht-ganzzahligen Vorwärtstransformation bildhaft dar. Ein transformierter Bildpunkt kann anteilig bis zu vier Pixel überdecken (von Wolberg als „*four-corner mapping paradigma*“ bezeichnet). Die neuen z-Werte ergeben sich aus der gewichteten Summe aller transformierten z-Werte, wobei der Anteil der Überdeckung eines Rasterpunkts als rationale Vorfaktoren addiert werden.

3.1.2 Gauß-Splatting

Ein zusätzlich auftretender Effekt ist die Nichtabdeckung von Rasterkoordinaten (Abbildung 3.4). Abhilfe schafft die Vorwärtstransformation mit Gauß-Splatting, d. h. dem räumlichen Verteilen (*splat*) des transformierten Rasterpunkts über ein größeres Gebiet entsprechend einer Normalverteilung [45].

Das Splatting-Verfahren erweitert die allgemeine Vorwärtstransformation und beruht ebenfalls auf der Addition von z-Werten mit Vorfaktoren im Zielraster. Im Gegensatz

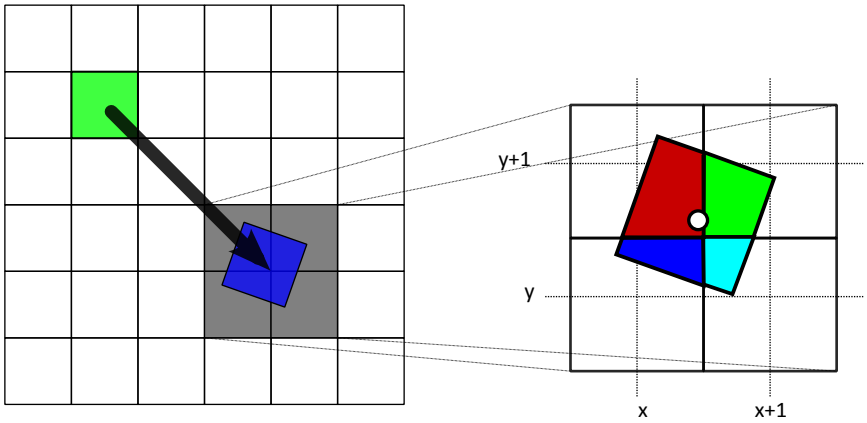


Abbildung 3.3: Vorwärtstransformation von gerasterten Messpunkten. Die Vorfaktoren der Tiefenwerte für jeden abgedeckten Rasterpunkt entsprechen dabei den anteiligen Flächen der transformierten Rasterpunkte an den um das Zentrum der Transformation (rot) herumliegenden Punkten.

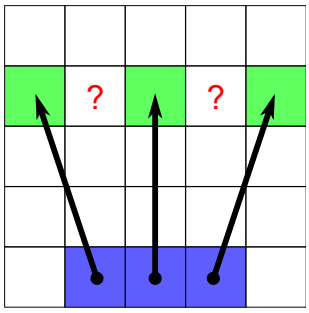


Abbildung 3.4: Vorwärtstransformation innerhalb eines Rasters mit unerreichten Rasterpunkten.

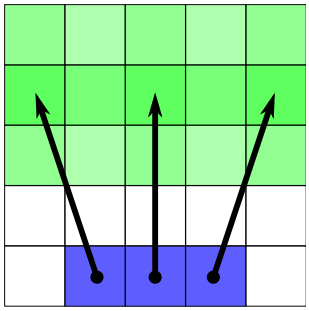


Abbildung 3.5: Gauß-Splatting bei der Vorwärtstransformation

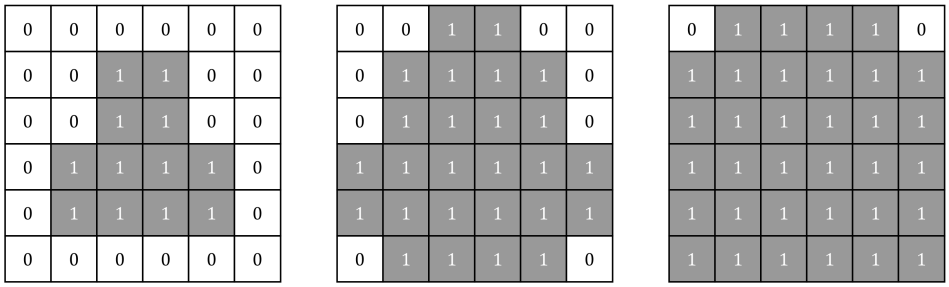


Abbildung 3.6: Binärbild-Dilatation mit zwei Schritten bei einer 4er-Nachbarschaft (d. h. oberer, unterer, linker oder rechter Nachbar = 1)

zur direkten Verschiebung erfolgt die Abdeckung der neuen Rasterpunkte anhand einer Verteilungsfunktion. Der Anteil des überdeckten Rasterpunkts entspricht in diesem Fall dem Wert der Normalverteilung bezogen auf den Abstand zum nächsten nicht-ganzzahligen Punkt.
Es funktioniert nach dem folgenden Schema:

1. Gegeben ist ein $m \times n$-Raster $T := \{t_{i,j}\} = \{z(\mathbf{p_{i,j}})\}$ mit $i = 0, \dots m-1$, $j = 0 \dots n-1$, für die Abbildung der zweidimensionalen Messpunkte $\mathbf{p_{i,j}}$ auf ihren Tiefenwert.

2. Initiiere ein $m \times n$-Raster $Z := \{z_{i,j}\}$ mit leeren Tiefenwerten, d. h. $\forall \mathbf{p_{i,j}} : z_{i,j} := 0$, und ein Raster $G := \{g_{i,j}\}$ für die Gauß-Vorfaktoren mit $\forall \mathbf{p_{i,j}} : g_{i,j} := 0$.

3. Berechne zu jedem Messpunkt $\mathbf{p}$ den Zielpunkt $\mathbf{q}$ der Transformation:
   - Sei $\mathbf{q_{k,l}} := \left(\lfloor q_s + \frac{1}{2} \rfloor, \lfloor q_t + \frac{1}{2} \rfloor\right)$ der nächste ganzzahlig gerundete Messpunkt mit der Rasterkoordinate (k, l) zu $\mathbf{q} = (q_s, q_t)$.
   - Bestimme die (Gauß-)Funktion $g(\mathbf{q}, \mathbf{q_{k,l}}) := e^{-\frac{1}{2}\sqrt{(x(\mathbf{q})-k)^2+(y(\mathbf{q})-l)^2}}$.
   - Addiere den gewichteten Tiefenwert $z(\mathbf{p})$ auf das bestehende Raster Z, d. h. $z_{k,l} \mathrel{+}= g(\mathbf{q}, \mathbf{q_{k,l}}) \cdot z(\mathbf{p})$.
   - Addiere den Faktor $g(\mathbf{q}, \mathbf{q_{k,l}})$ auf das Raster G, d. h. $g_{k,l} \mathrel{+}= g(\mathbf{q}, \mathbf{q_{k,l}})$.

4. Skaliere das Raster Z mit $\frac{Z}{G}$, d. h. $z_{i,j} := \frac{z_{i,j}}{g_{i,j}}$.

Dies ergibt die neuen Tiefenwerte der transformierten Messpunkte im Raster Z.

3.1.3 Dilatation

Die Dilatation (*dilation*) nach Serra ist ein morphologischer Operator aus der Bildverarbeitung [78]. Dieser ist in der Lage, innerhalb eines Binärbilds mit 0-1-Pixeln (d. h. nicht-vorhanden / vorhanden, maskiert / unmaskiert) einen ausgewählten Bildbereich sukzessiv zu vergrößern. Jede Iteration setzt genau die 0-Pixel auf 1, die mindestens einen 1-Pixel als Nachbarn besitzen (Abbildung 3.6). Somit wächst das maskierte Gebiet. Wiederholtes Durchführen des Operators führt schließlich zu einer vollständigen Maskierung des gesamten Gebiets.

Der Grundgedanke der Dilatation wird aufgegriffen, um unbekannte Gebiete einer Bathymetrie (d. h. Rasterpunkte ohne bekannten Tiefenwert) anhand der bekannten

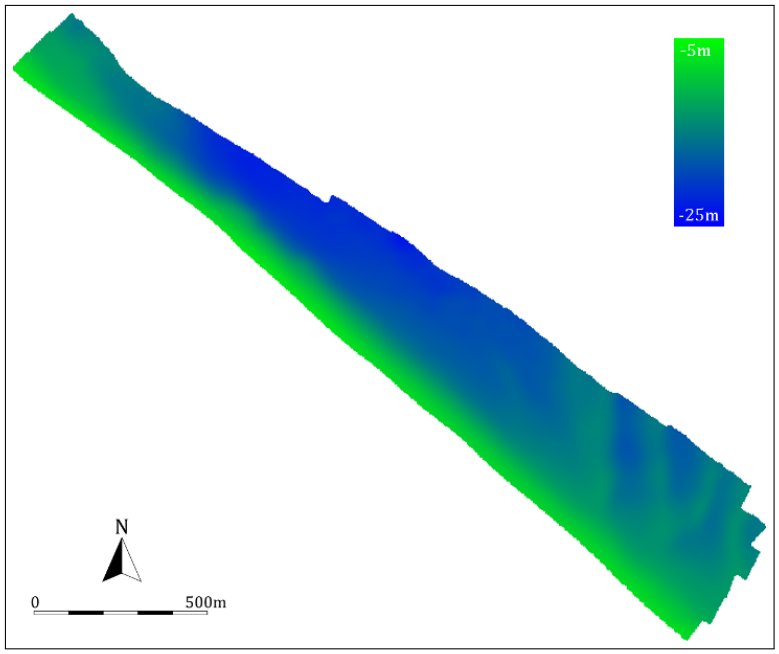


Abbildung 3.7: Die Beispieldaten überdecken eine Grundfläche von 2249 m × 1865 m bei einem Rasterabstand von 1 m. Die dargestellte Skalierung der Tiefenwerte liegt zwischen -25 m (grün) und -5 m (blau).

Gebiete zu besetzen. Dieser Schritt dient dazu, eine vollbesetzte und kontinuierliche Bathymetrie zu erzeugen. Derartige Tiefenwerte besitzen allerdings keine Konfidenz, d. h. ihr Wahrheitsgehalt und ihre Praktikabilität sind nicht gegeben.

3.2 Direkte Reduktion

Die direkt ausgeführte Reduktion bathymetrischer Geländemodelle beruht auf

- dem Import und der Verrasterung einer Bathymetrie als Referenzgeometrie
- der T-Spline-Flächen-basierten Approximation eines T-Netzes an die Referenzbathymetrie
- dem Export des T-Netzes.

Der allgemeine Ablauf wird für an einem Beispieldatensatz vorgeführt. Abbildung 3.7 zeigt die grafische Darstellung der Daten. Es handelt sich um eine einzelne Teilkarte einer großflächigen Vermessung in der Bremer Bucht (Kapitel 7). Das rechteckige Gesamtgebiet umfasst einen Bereich von 2249 m × 1865 m und die Tiefe variiert zwischen -23,6 m und -9,2 m. Der Abstand der Rasterpunkte beträgt 1 m.

3.2.1 Import und Verrasterung

Sowohl Primärdaten, d. h. punktuelle Messdaten im Einzugsgebiet, als auch verarbeitete Sekundärdaten stellen mögliche Datenquellen für die vorgestellte Reduktion dar. Die nachfolgenden Operationen müssen allerdings unabhängig von den ursprünglichen Datenformaten erfolgen. Aus diesem Grund werden die genutzten Daten vorab in ein einheitliches Datenschema konvertiert. Die Wahl fällt hierbei auf hochaufgelöste Rasterdaten für Bathymetrien aufgrund der folgenden Vorteile:

- Eine **effiziente Speicherung** der Rastergeometrie: Die Position eines Punkts im Raster induziert seine geometrische Koordinate. Diese ergibt sich aus der Definition eines Anfangspunkts (x_0, y_0) und den Abständen der Rasterpunkte d_x und d_y untereinander (falls abweichend von 1 m). Jede Rasterposition enthält lediglich einen Tiefenwert oder einen speziellen NOVALUE-Wert[1] (z. B. eine Not-a-Number-Gleitkommazahl – `NaN`).

- Mit gängigen Verfahren der **Transformation und Kompression** von Bilddaten können die Rasterdaten effektiv verarbeitet werden [5]. Eingesetzt werden hierbei oftmals die verlustbehaftete Kosinus-Transformation (angelehnt an das Bilddatenformat der *Joint Photographic Experts Group*, JPEG) oder die Wavelet-Transformation mit variablen Kompressionsraten [79].

- Tiefenwerte an nicht-ganzzahligen Zwischenpositionen werden mit bilinearer oder polynomieller **Interpolation** direkt aus den umliegenden Werten ermittelt. Extrapolationen sind lediglich an nicht besetzten Positionen mit unbekannten NOVALUE-Werten notwendig.

- Die oben aufgelisteten Eigenschaften werden direkt in der **GPGPU**-Programmierung unterstützt (Kapitel 6).

Liegt der Abstand der Rasterpunkte unterhalb oder nahe an der Dichte der originalen Messdaten, so ist nur ein geringer qualitativer Verlust zu erwarten. Im betrachteten Beispiel handelt es sich bereits um die interpolierte und gerasterte Konvertierung von gefilterten Primärdaten aus einem nicht näher bekannten Format. Diese Daten sind somit als Sekundärdaten zu betrachten und werden ohne weitere Konvertierung verwendet. Liegen die Daten in einem anderen Datenformat vor, so wird mittels eines beliebigen Interpolationsverfahrens ein neues Raster erzeugt und als solches gespeichert.

T-Spline-Flächen werden im Verlauf der Approximation laufend mit der Referenzbathymetrie verglichen. Dies erfordert daher eine effiziente Interpolation der Originaldaten. Als Ergebnis der Verrasterung existiert somit ein sehr großer Datensatz, der in der vorgestellten Anwendung deutlich mehr Daten- als ursprüngliche Messpunkte besitzt. Letztlich rechtfertigt die Verrasterung durchaus den erhöhten Speicherbedarf gegenüber dem zusätzlichen Rechenaufwand für die Interpolation der Originaldaten. Nicht zuletzt sind die Rasterdaten direkt in GPU-optimierte Bilddaten, z. B. als Datentyp `image2d` (Abschnitt 6.1.5), konvertierbar.

3.2.2 Approximation und Flächenrückführung

Der globale Prozess der Reduktion benötigt eine frei wählbare Approximationsgüte. Diese drückt generell das Maß der Optimierung aus, d. h. die Abbruchbedingung des Approximationsschemas. Mögliche Strategien innerhalb der Flächenrückführung wären:

- das Unterschreiten einer punktuell maximalen Differenz der Tiefenwerte

- das Unterschreiten eines durchschnittlichen Differenzfehlers

- das Erzwingen von geometrischen Randbedingungen, z. B. die Einhaltung der maximalen Krümmung oder die Beachtung der Neigung von Teilflächen

[1] In Anlehnung an das ESRI-GIS-Datenformat.

- das Erreichen einer maximalen Anzahl an Kontrollpunkten oder Zellen im T-Netz
- eine vorgegebene Anzahl an Iterationen unabhängig von der erreichten Güte
- beliebige Kombinationen der oben genannten Ziele.

Piegl und Tiller beschreiben ein allgemeines Approximationsschema für die Flächenrückführung ausgehend von einer Referenzfläche (Algorithmus 1) [66]. Hierzu verwenden sie ein initiales Kontrollpunktnetz mit einer minimal notwendigen Anzahl an Kontrollpunkten (z. B. 3×3). Die Autoren führen in ihrem Schema lediglich die einzelnen Teilschritte auf, nennen allerdings keine konkreten Methoden für die Teilschritte.

Eingabe: Ein initiales Netz mit minimaler Anzahl an Kontrollpunkten
begin
  – Erzeuge eine geometrische Fläche aus dem Kontrollpunktnetz
  repeat
    – Verfeinere die Fläche
    – Optimiere die Kontrollpunkte im Netz anhand der Referenzfläche
    – Bestimme den Fehler als Gütekriterium zwischen der erzeugten Fläche und der Referenzfläche
  until *Abbruchkriterium erfüllt*;
end
Ausgabe: Ein optimiertes Kontrollpunktnetz der Fläche

Algorithmus 1 : Allgemeine modulare Bottom-Up-Approximation nach [66]

Darauf aufbauend stellt das nachfolgende Kapitel 4 ein Approximationsschema für T-Spline-Flächen vor, das ein T-Netz erzeugt und progressiv verfeinert. Hierfür werden eine wählbare, maximale Fehlerdistanz zwischen der Bathymetrie und der T-Spline-Fläche, sowie der Grad K vorgegeben. Das Ergebnis ist ein T-Netz, das entsprechend der Approximationsgüte eine T-Spline-Fläche modelliert, um die Referenzfläche anzunähern.

3.2.3 Export und Maskierung

Das T-Netz stellt das Resultat und den Export des Approximationsschemas dar (das finale T-Netz ist dargestellt in Abbildung 3.12). Eine Bathymetrie ist i. A. nicht lückenlos, d. h. die verwendeten Rasterdaten sind nicht vollständig belegt und / oder enthalten Lücken. Ein erzeugtes T-Netz hingegen umfasst einen rechteckigen Parameterbereich, und es deckt prinzipiell nicht besetzte Randgebiete ab. Die Adaptivität des T-Netzes repräsentiert diese Gebiete allerdings durch deutlich größere Zellen.

Eine finale, jedoch optionale, Maskierung schneidet die T-Spline-Fläche entsprechend der originalen Bathymetrie zurecht. Ein Flächenpunkt der T-Spline-Fläche $\mathbf{t}(s,t)$ liegt innerhalb der finalen Fläche, falls sich zu diesem ebenfalls ein Punkt in der (verrasterten) Bathymetrie wiederfindet. Die Maskierung erfolgt konzeptbedingt als letzter Schritt der Reduktion. Daher ist eine zusätzliche Bitmaske (Wert vorhanden / nicht vorhanden) in der gleichen Auflösung des Rasterdatensatzes notwendig und die Bathymetrie wird entsprechend kodiert. Vorhandene Tiefenwerte werden

auf 1 gesetzt, nicht vorhandene (*NOVALUE*) auf 0. Gängige Algorithmen für die Binärraster- bzw. Bitmasken-Komprimierung erreichen sehr gute Effizienz[2].

3.2.4 Grenzen der Optimierung

In Abschnitt 4.2.1 wird das Optimierungsverfahren für die Methode der kleinsten Quadrate erläutert. Das dort im Detail beschriebene Verfahren beruht auf einer Optimierungsmatrix, deren Größe sich aus der Anzahl der Kontrollpunkte im T-Netz und aus der Anzahl der verwendeten Messpunkte für die Bathymetrie ergibt. Die Dimension der Matrix T aus Formel (4.35) ist wie folgt definiert: die Reihendimension in T entspricht der Anzahl der Messpunkte aus der Bathymetrie, und die Spaltendimension entspricht der Anzahl der Kontrollpunkte im zu optimierenden T-Netz.

Betrachtet man eine feste Anzahl an Messpunkten (in jeder Iterationsstufe können die gleichen Messpunkte aus der ursprünglichen Bathymetrie wiederverwendet werden), so wächst die Größe der Optimierungsmatrix in jeder Iteration abhängig von der Anzahl der im T-Netz neu erzeugten Kontrollpunkte.

Gibt man eine maximale Elementanzahl für die Optimierungsmatrix T vor (beispielsweise bedingt durch die technischen Grenzen der Lösungsbibliothek[3]), so ergibt sich die maximale Anzahl der zu optimierenden Kontrollpunkte als Anzahl der maximalen Matrixelemente (ganzzahlig) geteilt durch die Anzahl der Messpunkte.

Die Größe der erzeugten T-Netze kann folglich durch eine Verminderung der Anzahl der Messpunkte erhöht werden. Falls die Messpunkte aus einer gleichmäßigen Abtastung der Bathymetrie (wie in dieser Arbeit) entstehen, so werden ihre Abstände zueinander größer. Hierbei wächst die Gefahr, dass die Abstände der neu eingefügten Kontrollpunkte im T-Netz in Einzelfällen unterhalb des kleinsten Abstands der Messpunkte fallen. Mit zunehmender Dichte des T-Netzes trifft dies auf immer mehr Kontrollpunkte zu.

Abbildung 3.13 zeigt das Resultat aus diesem Verhalten anschaulich. Die Ursache hierfür ist das Unterschreiten der Abstände der Kontrollpunkte im Vergleich zu den Abständen der Messpunkten. Im betrachteten Beispiel liegt der minimale Abstand der Kontrollpunkte im T-Netz bei 2,72 m, der Abstand der Messpunkte aus der Bathymetrie allerdings bei 3 m. Infolgedessen ist das überbestimmte Gleichungssystem in Gleichung (4.33), bezogen auf einzelne Kontrollpunkte, unterbesetzt, d. h. die Berechnung des Tiefenwerts an einem Kontrollpunkt greift auf nicht ausreichend viele oder keine gemessenen Tiefenwerte zurück. Eine optimale, globale Lösung ist somit nicht berechenbar. Daher ist die Bewahrung einer Untergrenze für die Abstände der Kontrollpunkte im T-Netz (d. h. für die Kantenlängen) zu beachten: der minimale Abstand der Kontrollpunkte darf nicht unterhalb des Abstands der Messpunkte fallen.

3.2.5 Auswertung des Beispiels

Der Prozess erzeugt auf fortlaufenden Verfeinerungsstufen T-Netze mit zunehmender Approximationsqualität für kubische T-Spline-Flächen (Abbildungen 3.8 bis 3.12). Ein jeweiliges Bild für die Klassen der Differenzfehler (zwischen 1 cm und 2 m) macht jeweils die fortlaufend abnehmenden Gebiete mit hohen Abweichungen zwischen der erzeugten

[2] Siehe z. B. [94]. Die Autoren geben allerdings keine Kompressionsraten an, da ihr Fokus auf dem Durchsatz liegt.

[3] Die verwendete Scilab-API erlaubt z. B. nur Matrizen mit weniger als 2^{31} Elementen (ca. 2 Milliarden).

Fläche und der Bathymetrie sichtbar. Es visualisiert somit die Approximationsgüte und stellt die (noch) fehlerbehafteten Bereiche dar.

Der Vergleich der Kenngrößen für die direkte Reduktion mit linearer Optimierung zeigt deutlich die Verbesserung bei zunehmender Dichte der Kontrollpunkte (Abbildung 3.14). Ein Histogramm verdeutlicht dies ebenfalls für einzelne Fehlerklassen (Abbildung 3.15).

In dem Beispiel sind folgende Artefakte deutlich erkennbar:

- Eine adaptive, treppenförmige Netzverfeinerung, insbesondere im Randbereich des Messgebiets.
- Daraus folgend eine ungleichmäßige Abdeckung des Messgebiets einschließlich der überhängenden Abdeckung von unerfassten Gebieten.
- Kontrollpunkte sind aufgrund der Topologie der T-Netze ungleichmäßig verteilt. Die lokalen Knotenwerte für die T-Spline-Basisfunktionen variieren somit und letztlich führt dies zu einer starken Variabilität im Einfluss von Kontrollpunkten.

Das T-Netz überragt die räumliche Ausdehnung der Bathymetrie zu jeder Seite um 25%, Details hierzu folgen in Abschnitt 4.1.1. Weiterhin fällt auf, dass das erzeugte T-Netz einen großen Teil einer Bathymetrie abbildet, der in den Messdaten nicht enthalten ist (ohne die erwähnte Überdeckung). Die zusätzlichen Flächenteile werden mit einer geringen Anzahl an Kontrollpunkten modelliert, insbesondere entstehen somit Teilgebiete ohne nennenswerte Bedeutung für das vermessene Kerngebiet. Das naheliegende Ziel soll sein, eine Bathymetrie durch Verschiebungen zu transformieren, um die rechteckige Grundform des T-Netzes abzudecken. Dadurch wird ein zusätzlicher Zwischenschritt motiviert, der die Originalbathymetrie durch globale und lokale Operationen bestmöglich in die gewünschte Form transformiert. Durchaus erwünscht sind Geländemodelle mit gleichmäßig strukturierten Kontrollpunktnetzen, und somit gleichmäßig verteilten Kontrollpunkten, so dass ein optimiertes T-Netz ebenfalls mit einer „harmonischen[4]“ Struktur einhergeht.

3.3 Indirekte Reduktion

Im vorherigen Abschnitt ist die direkte Reduktion eines bathymetrischen Geländemodells beschrieben, d. h. die T-Spline-Fläche bildet direkt die Bathymetrie ab. Ergänzende Schritte kapseln den bisherigen Ablauf innerhalb einer hin- und einer rückführenden (d. h. inversen) Transformation der Bathymetrie.

[4] Im Sinne einer minimierten Anzahl an inneren Kontrollpunkten mit T-Kreuzungen, d. h. nahe an einer optimalen Gitter-Struktur

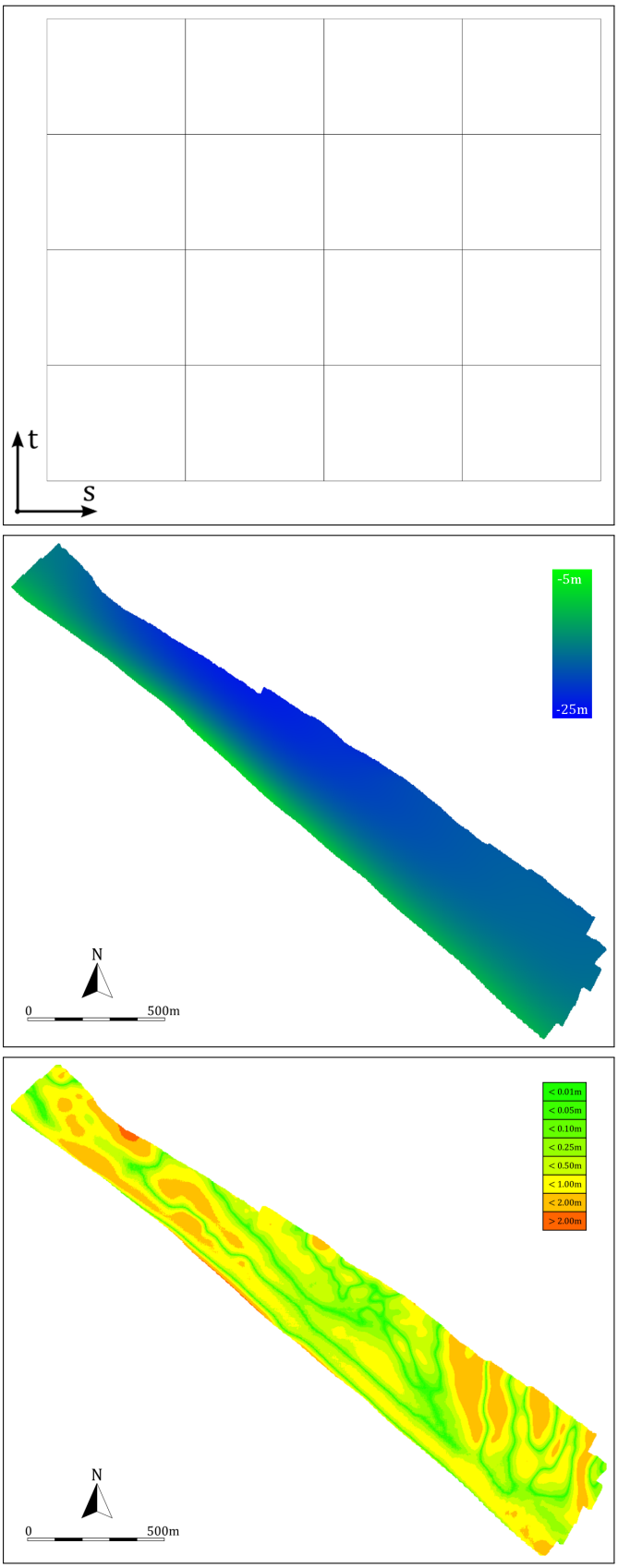


Abbildung 3.8: Initiales T-Netz, kubische T-Spline-Fläche und Klassifikation der Differenzfehler

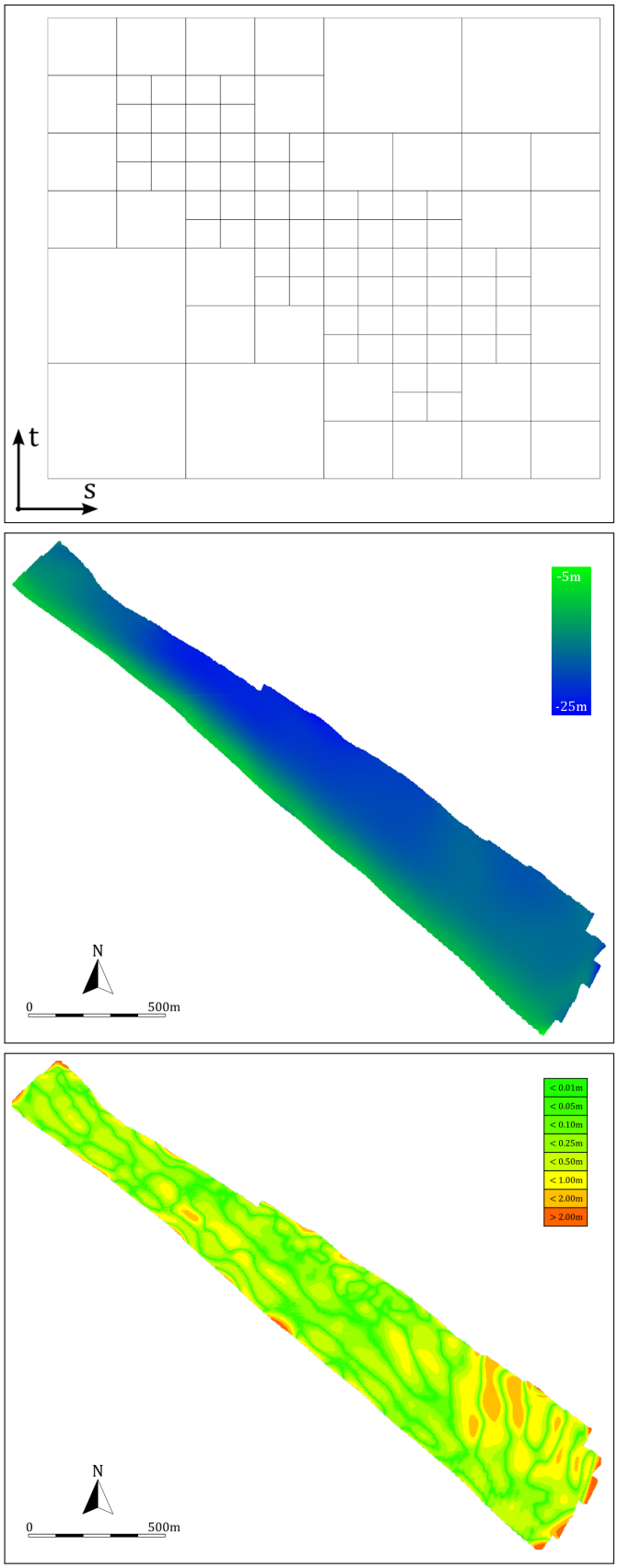


Abbildung 3.9: T-Netz nach 2. Iteration, kubische T-Spline-Fläche und Klassifikation der Differenzfehler

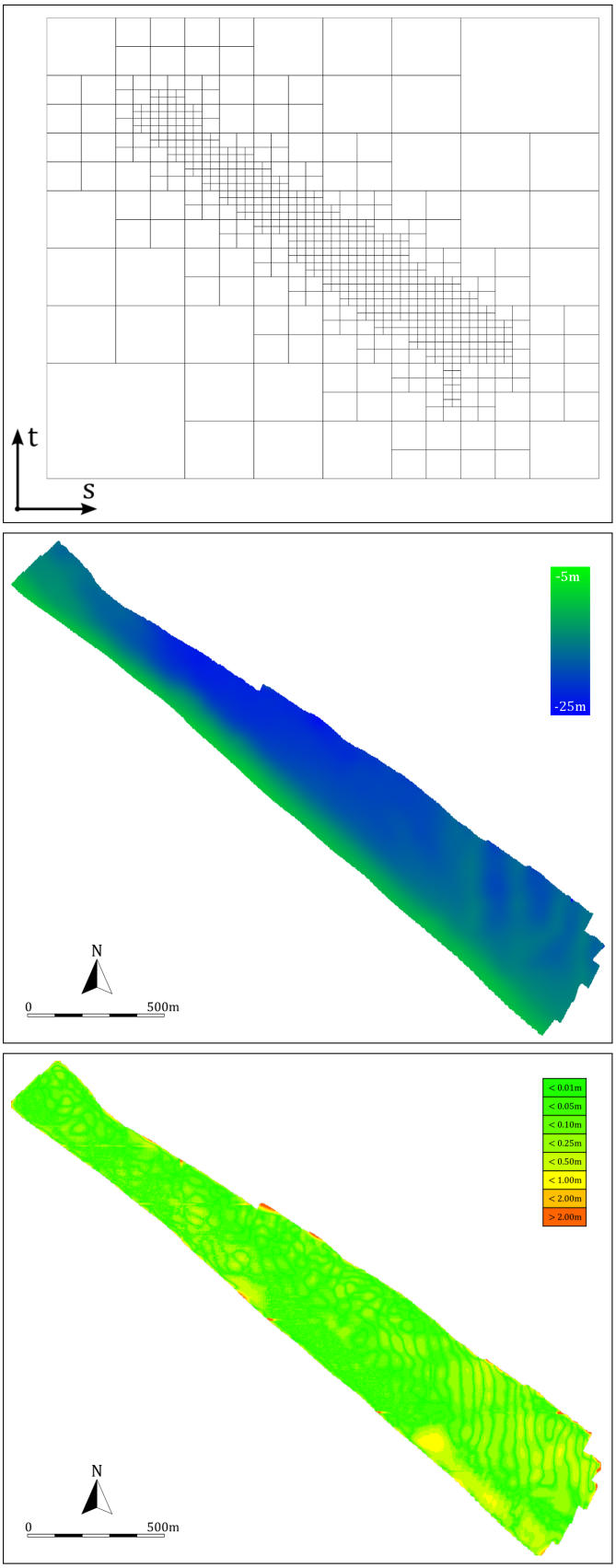


Abbildung 3.10: T-Netz nach 4. Iteration, kubische T-Spline-Fläche und Klassifikation der Differenzfehler

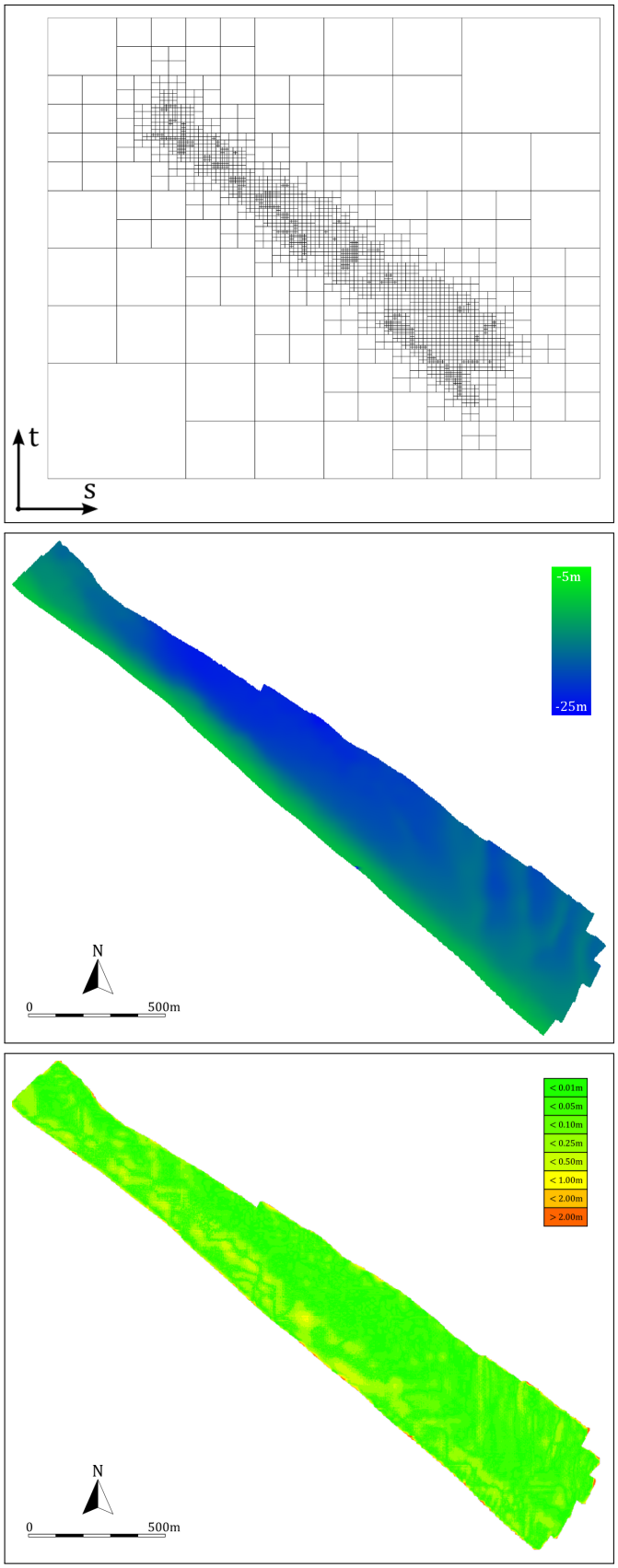


Abbildung 3.11: T-Netz nach 6. Iteration, kubische T-Spline-Fläche und Klassifikation der Differenzfehler

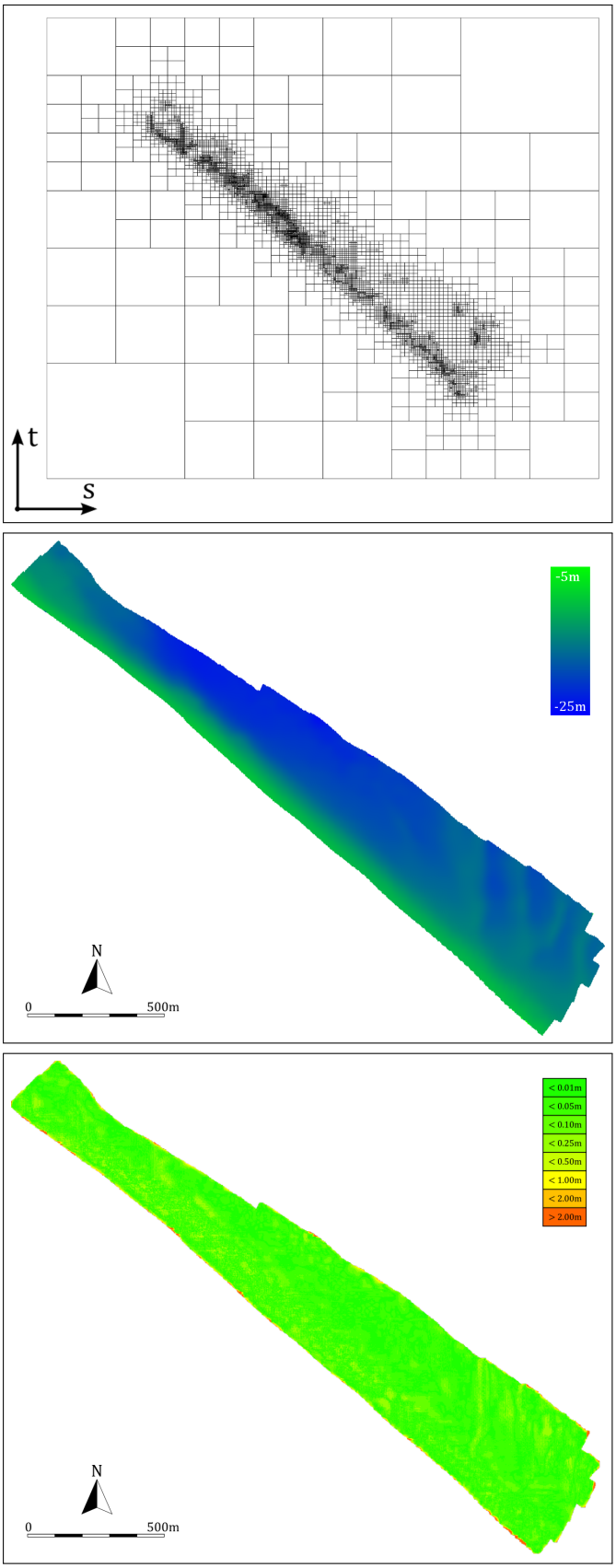


Abbildung 3.12: T-Netz nach 8. Iteration, kubische T-Spline-Fläche und Klassifikation der Differenzfehler

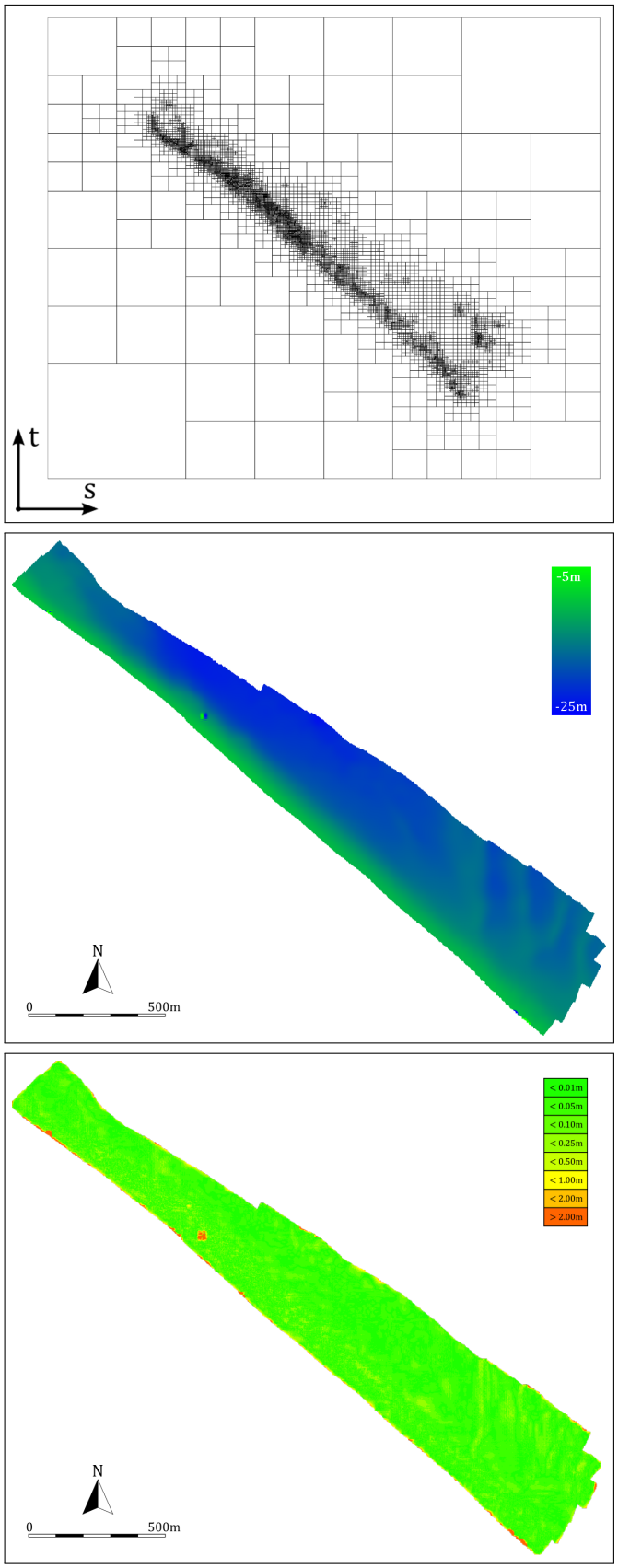


Abbildung 3.13: T-Netz nach 9. Iteration, kubische T-Spline-Fläche und Klassifikation der Differenzfehler. Deutlich sichtbar sind die Approximationsfehler bedingt durch die zu geringen Kontrollpunktabstände im T-Netz.

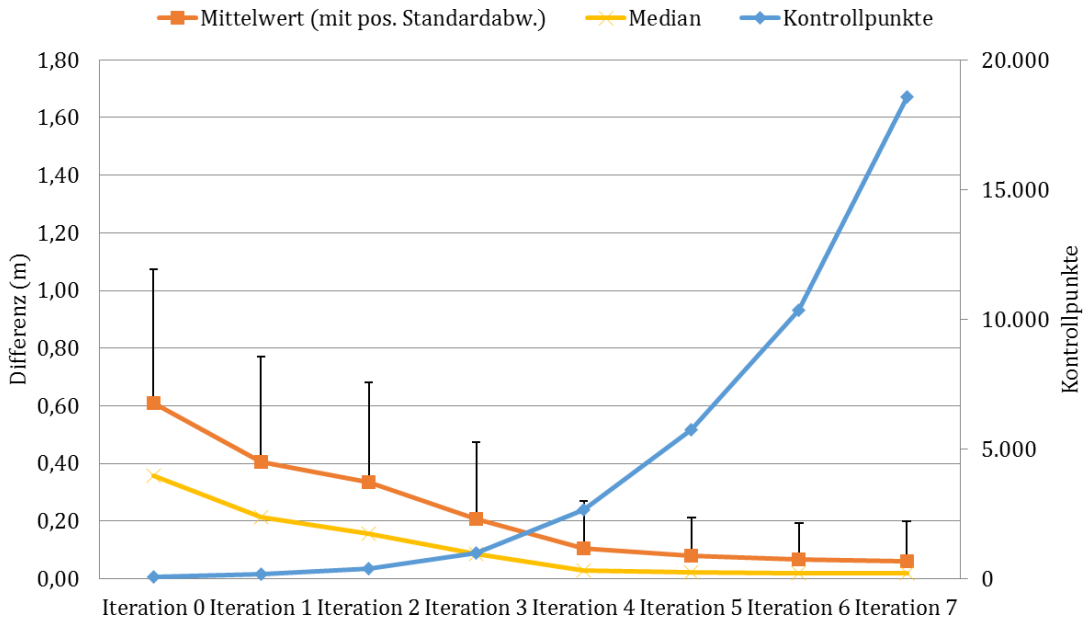


Abbildung 3.14: Vergleich der Qualität der direkten Optimierung bei zunehmender Anzahl an Kontrollpunkten. Pro Stufe sind dies der Mittelwert (mit positiver Standardabweichung) und Median der Differenzfehler, zusätzlich die Anzahl der Kontrollpunkte als Indikator der Netzgröße.

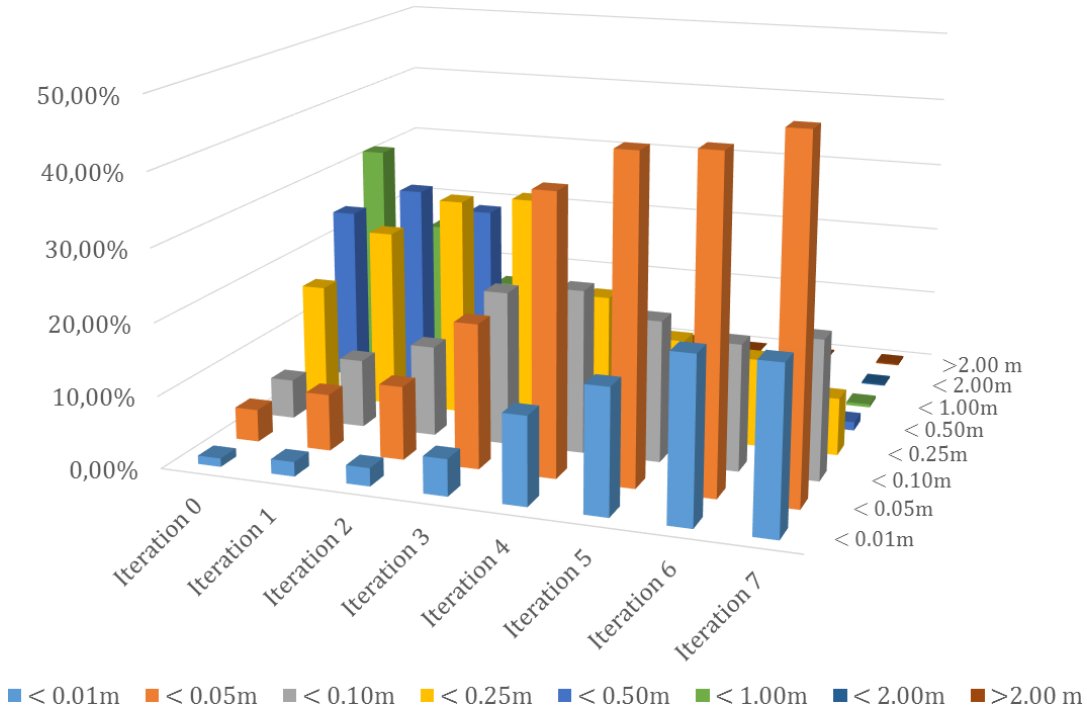


Abbildung 3.15: Histogramm der Differenzfehler der direkten Optimierung

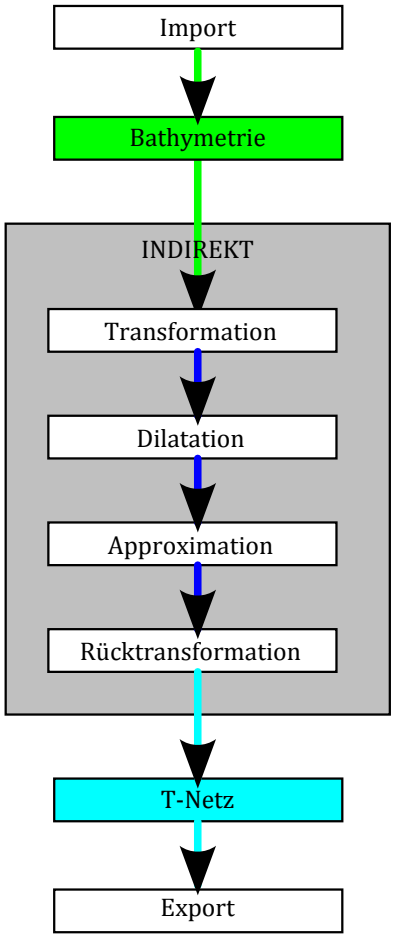


Abbildung 3.16: Darstellung der indirekten Reduktion

Der erweiterte, indirekte Prozess besteht somit aus den folgenden Schritten (Abbildung 3.16):

1. Import und Verrasterung einer Referenzbathymetrie
2. Kombinierte Transformation der Bathymetrie mittels einer globalen Rotation und lokalen Transformationen (*warping*).
3. Randapproximation mittels Dilatation
4. T-Spline-Flächen-basierte Approximation eines T-Netzes an die Referenzbathymetrie
5. Invertierte Transformation aus Schritt 2
6. Export des T-Netzes.

Die Beschreibung der detaillierten Abläufe der kombinierten Transformation erfolgt in Abschnitt 3.3.2. Diese erzeugt als zusätzliche Information ein Verschiebungsfeld mit einer frei wählbaren Anzahl an Verschiebungspunkten[5]. Ein solches Verschiebungsfeld

[5] In der Bildverarbeitung oft als *landmark* bezeichnet. Sie treten als markante Punkte innerhalb eines Bildes hervor und sind somit leicht detektierbar. In dieser Arbeit sind die Verschiebungspunkte allerdings durch feste Positionen in der Bathymetrie bestimmt.

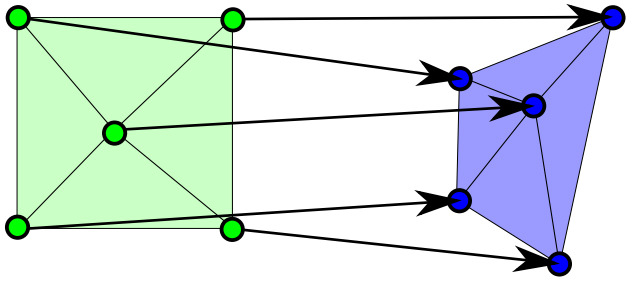


Abbildung 3.17: Darstellung eines einfachen, invertierbaren Verschiebungsfeld

$W := \{(\mathbf{x_i}, \boldsymbol{\delta_i})\}$ entspricht einer Menge an Verschiebungspunkten $\mathbf{x_i} \in \mathbb{R}^2$ mit einem zugehörigen Verschiebungsvektor $\boldsymbol{\delta_i} \in \mathbb{R}^2$. Das Verschiebungsfeld gibt die neuen Positionen der Messpunkte nach der Verschiebung vor (Abschnitt 3.3.2).

Ein zusätzlicher Schritt der Dilatation füllt einen Rasterdatensatz durch Extrapolation auf, bis ein vollständig beschriebener, rechteckiger Rasterdatensatz entsteht (Abschnitt 3.3.3). Letztlich stellt dieses vollständig bekannte Raster die Referenz für die folgende Berechnung einer T-Spline-Fläche dar. Der Ablauf hierfür entspricht exakt der direkten Reduktion (Abschnitt 3.2.2), an dieser Stelle allerdings auf dem transformierten und erweiterten Datensatz.

Die beiden finalen Schritte überführen das T-Netz für die transformierte Bathymetrie in ein T-Netz für die Originalbathymetrie. Die Invertierung der kombinierten Transformation ist in Abschnitt 3.3.5 beschrieben, der Abschluss des Prozesses und der finale Export erfolgen analog zu den Arbeitsschritten der direkten Reduktion (Abschnitt 3.2.3).

3.3.1 Import und Verrasterung

Der Ablauf ist identisch mit dem Einstieg der direkten Reduktion (Abschnitt 3.2.1).

3.3.2 Globale und lokale Transformation

Die Transformation der importierten Messpunkte erfolgt in zwei kombinierten Schritten. Der Zweck dieser Operation besteht darin, die Messdaten global auszurichten und gleichzeitig eine innere Ausdehnung der Rastergeometrie im Messgebiet zu erreichen. Somit entsteht ein zusammenhängender Rasterdatensatz für die Bathymetrie als Eingabe für das nachfolgende Approximationsschema (Abschnitt 3.3.4).

Der vorgestellte Ansatz berechnet die kombinierte globale und lokale Transformation für eine Bathymetrie und gibt ein Verschiebungsfeld vor. Mittels bivariater Dreiecksinterpolation zwischen den Verschiebungspunkten werden für einzelne Rasterpunkte die Verschiebungen und somit die neuen Rasterkoordinaten bestimmt.

Globale Rotation

Die Hauptachsentransformation[6] ist (im Zusammenhang mit dieser Arbeit) ein Werkzeug zur Identifikation der Charakteristika von Flächen oder Körpern. In der Mechanik wird sie für die Bestimmung von physikalischen Eigenschaften von starren Körpern

[6] Diese wird auch Hauptkomponentenanalyse (*principal component analysis* – PCA) oder Karhunen-Loève-Transformation genannt.

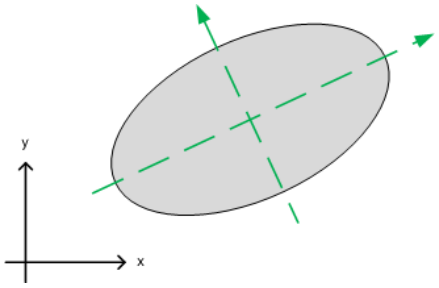


Abbildung 3.18: Zweidimensionale Ellipse mit Hauptachsen (grün)

verwendet [88]. Darüber hinaus findet sich die Hauptachsentransformation in der quantitativen Forschung zur Reduktion von stochastischen Datensätzen wieder: hier werden (scheinbar) zufällig verteilte Messpunkte aus dem höheren Raum der Erhebungsgrößen auf einen Raum mit kleinerer Dimension projiziert, um Gesetzmäßigkeiten zu entdecken.

Die Hauptachsen definieren somit ein lokales, orthogonales Koordinatensystem, das die Lage und Orientierung eines Körpers in seinem umgebenden Raum vorgibt (Beispiel einer Ellipse in Abbildung 3.18).

Betrachtet man ein computergrafisches Bild aus monochromen Pixeln als zusammenhängende Fläche und interpretiert die Grauwerte jedes Pixels (zwischen 0 und 1) als Masse eines Teilstücks der Fläche, so ist die Hauptachsentransformation ebenfalls durchführbar. Auf diesem Weg wird die Orientierung von prägnanten Merkmalen in Bildern erkannt (z. B. die Ausrichtung von Computertomographien) [4].

In dieser Arbeit wird die Hauptachsentransformation benutzt, um die räumliche Ausdehnung einer vermessenen Bathymetrie zu bestimmen. Dies wird durch die Verwendung eines Rasterdatensatzes für die Bathymetrie erreicht. Die Messdaten liegen weiterhin im dreidimensionalen Raum, sie werden aber durch eine starre, zweidimensionale Scheibe mit einer Massenverteilung entsprechend der Tiefenwerte repräsentiert. Tatsächlich drückt der Tiefenwert an einer Stelle des Rasters bereits die Masse der auf dem Punkt der Bathymetrie ruhenden Wassersäule aus[7].

Die Berechnung der Hauptachsen erfolgt über die Momente einer zweidimensionalen Fläche. Allgemeine zweidimensionale Momente M_{mn} der Ordnung $m+n$ für Rasterpunkte $(x, y, z(x, y))$ sind definiert als

$$M_{\mathrm{mn}} := \sum_x \sum_y x^m \cdot y^n \cdot z(x, y). \tag{3.1}$$

Für $m = n = 0$ entspricht das Moment der Summe der Tiefenwerte im Raster:

$$M_{00} = \sum_x \sum_y z(x, y). \tag{3.2}$$

Die Momente erster Ordnung sind somit

$$\begin{aligned} M_{10} &= \sum_x \sum_y x \cdot z(x, y) \\ M_{01} &= \sum_x \sum_y y \cdot z(x, y). \end{aligned} \tag{3.3}$$

[7] Verständlicherweise ist der Tiefenwert selbst eine positive Zahl für negative z-Werte.

Der zweidimensionale Schwerpunkt $\mathbf{s}$ der Bathymetrie liegt bei

$$\mathbf{s} = (x_s, y_s) := \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right). \tag{3.4}$$

Für die Berechnung der Rotation ist nun die Trägheitsmatrix aus den Momenten zweiter Ordnung zu bestimmen. Diese beziehen sich auf den Schwerpunkt, so dass eine Koordinatentransformation der Rasterdaten durchzuführen ist. Die zentralen Momente $\hat{M}_{\mathrm{mn}}$ zweiter Ordnung sind nun definiert als

$$\begin{aligned} \hat{M}_{11} &:= \sum_x \sum_y (x - x_s) \cdot (y - y_s) \cdot z(x, y) \\ \hat{M}_{20} &:= \sum_x \sum_y (x - x_s)^2 \cdot z(x, y) \\ \hat{M}_{02} &:= \sum_x \sum_y (y - y_s)^2 \cdot z(x, y). \end{aligned} \tag{3.5}$$

Die Trägheitsmatrix D für die zweidimensionale Hauptachsentransformation lautet

$$D := \frac{1}{M_{00}} \begin{pmatrix} \hat{M}_{20} & \hat{M}_{11} \\ \hat{M}_{11} & \hat{M}_{02} \end{pmatrix}. \tag{3.6}$$

Gesucht ist nun die Rotationsmatrix R aus der Eigenwertzerlegung der quadratischen Matrix D:

$$D = R^{-1} T R. \tag{3.7}$$

Die Rotationsachsen der Bathymetrie entsprechen den Eigenvektoren aus den Spalten der (quadratischen) Matrix R, und T ist eine Diagonalmatrix mit den Eigenwerten.

Besitzt das betrachtete Gebiet einen deutlich ausgeprägten geradlinigen Verlauf, beispielsweise aus einem Strömungsverlauf heraus entstanden, so ist die Rotation anhand dieser Orientierung durchführbar. Die primäre Hauptachse wird hier nun als diejenige definiert, die der hauptsächlichen Ausdehnung der Bathymetrie im Anwendungsfall entspricht, die sekundäre ist dazu orthogonal gerichtet. Anzumerken ist, dass die Rotation optional ist und bei mangelnder eindeutiger Ausprägung der Bathymetrie entfallen kann.

Der Winkel zwischen der Ost-West-Achse[8] und der primären Hauptachse des Gesamtgebiets gibt die globale Rotation vor. Der Rotationspunkt liegt im Schwerpunkt der Bathymetrie (Abbildung 3.19). Liegen die dominierende Achse des T-Netzes im Parameterraum und die Hauptachse des Gebiets im Geometrieraum übereinander, so ist prinzipiell eine bessere Approximation der T-Spline-Fläche an die Bathymetrie möglich, als in der direkten Reduktion (im Vergleich zu den Abbildungen 3.8 ff.). Eine Stufenbildung innerhalb des T-Netzes wird reduziert. Stehen die Achsen orthogonal zueinander, so ist das T-Netz um 90° zu drehen. Der Beispieldatensatz besitzt eine um etwa $30,1°$ gedrehte Orientierung zur Ost-West-Achse.

Eine alleinige Rotation der Bathymetrie garantiert allerdings nicht, dass die T-Netze die Randgebiete ausreichend gut abdecken. Hierbei stellen sich weiterhin deutliche Treppenartefakte ein (Abbildung 3.20), da das T-Netz und der Verlauf der Bathymetrie nicht immer im rechten Winkel zueinander stehen. Die globale Rotation führt

[8] Die Wahl der Achse erfolgt anhand des kleineren Winkels, diese kann prinzipiell auch Nord-Süd-Ausrichtung besitzen.

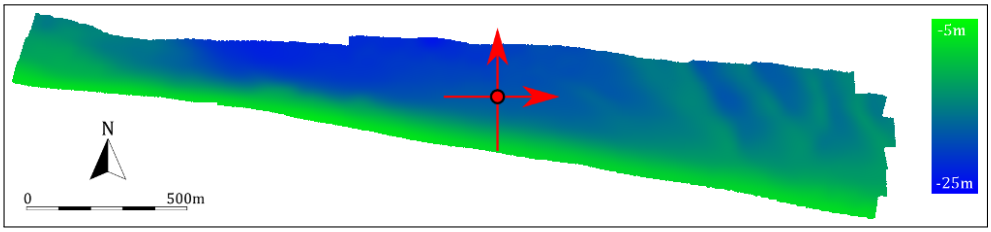


Abbildung 3.19: Bathymetrie mit Hauptachsen (rot) nach alleiniger Rotation um den Schwerpunkt

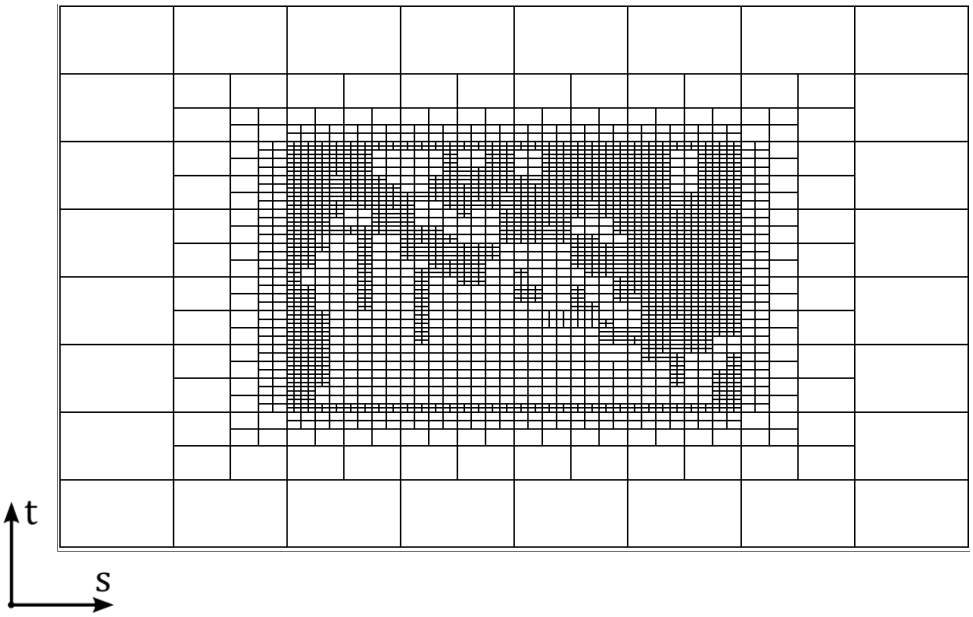


Abbildung 3.20: Darstellung einer möglichen T-Netz-Approximation nach alleiniger Rotation

erwartungsgemäß lediglich zu einer Verminderung des durchschnittlichen Winkels zwischen der dominierenden Richtung des T-Netzes und der Bathymetrie.

Die Notwendigkeit einer rasterbasierten Interpolation ergibt sich bereits aus der globalen Rotation. Diese bildet offensichtlich ganzzahlige Rasterkoordinaten auf nichtganzzahlige Koordinaten ab, und die somit entstehende Geometrie muss in ein bestehendes Raster eingepasst werden. Dies geschieht durch die sogenannte *Splatting*-Interpolation (Abschnitt 3.1.2).

Lokale Warping-Transformation

Ein Nachteil der globalen Rotation besteht darin, dass die resultierende Geometrie weiterhin die unregelmäßige Ausdehnung der Bathymetrie beibehält. Dies bedeutet, dass die betrachteten Gebiete stellenweise schmal verlaufen (z. B. Schifffahrtsrinnen, Kanäle), wiederum an anderer Stelle weiträumig gestreckt sind (Wattgebiete o. ä.). Ein weiterer Schritt passt nun die ursprüngliche Bathymetrie mit dem zu Beginn eingeführten Warping-Verfahren entsprechend der genannten Erfordernisse (raumfüllende, kontinuierliche Ausdehnung) an. Dieser Operator der Bildverarbeitung verzerrt die Bathymetrie (nach der Rotation) derart, dass diese eine regelmäßige und insbesondere flächendeckende Struktur ergibt. Die nachfolgende Approximation erstellt aus dieser „deformierten“ Fläche ein T-Netz mit einem reduzierten Treppeneffekt.

Lokale Transformationen beruhen auf Verschiebungspunkten mit Verschiebungsvektoren. Die Verschiebungspunkte liegen im Gebiet der Bathymetrie, bezeichnet als Urgebiet, und transformieren die Rastergeometrie in das Zielgebiet. Ein grobes, äquidistantes Raster (bezeichnet als Warping-Raster) aus Verschiebungspunkten deckt das Urgebiet ab (im Beispiel mit 20 m Abstand). Die Verschiebungspunkte werden mittels Bowyer-Watson-Algorithmus in eine Delaunay-Triangulation für die baryzentrische Dreieckinterpolation überführt [13].

Der umgesetzte Warping-Algorithmus besteht nun aus zwei Phasen: zunächst wird die x-Verschiebung blockweise und daran anschließend die y-Verschiebung bestimmt. Die komponentenweise Aufteilung ist notwendig, da die beschriebenen lokalen Verschiebungen nicht getrennt berechnet werden können. Die isolierten Verschiebungen an jedem Verschiebungspunkt addieren sich dann zu einem kombinierten Verschiebungsvektor, der in Verknüpfung mit der bekannten Rotation die kombinierte, globale Transformation in das Zielgebiet ergibt. Zu ergänzen ist, dass weitere Verschiebungspunkte am Rand der Bathymetrie hinzuzufügen sind, die allerdings nur durch die Rotation verändert werden. Dies stellt die Verschiebung der vollständigen Bathymetrie sicher.

Dieses Vorgehen ist in Abbildung 3.21 für das Beispiel aus Abbildung 3.7 dargestellt, zusätzlich wird die globale Rotation der Verschiebungspunkte durchgeführt. Abbildung 3.22 stellt die überlagerten Triangulationen von Ur- und Zielgebiet dar.

x-Verschiebung

Die Verschiebungspunkte sind entlang eines Gitters angeordnet, das sich parallel zu der Hauptachse der Bathymetrie erstreckt. Die x-Verschiebung bestimmt sich nach folgendem Schema:

1. Das Gitter wird in vertikale, streifenförmige Blöcke mit einer festen Breite l unterteilt, sie beträgt im Beispiel 50 m (d. h. das 2,5-fache des Abstands der Verschiebungspunkte). Die Gesamtbreite des Gitters beträgt L.

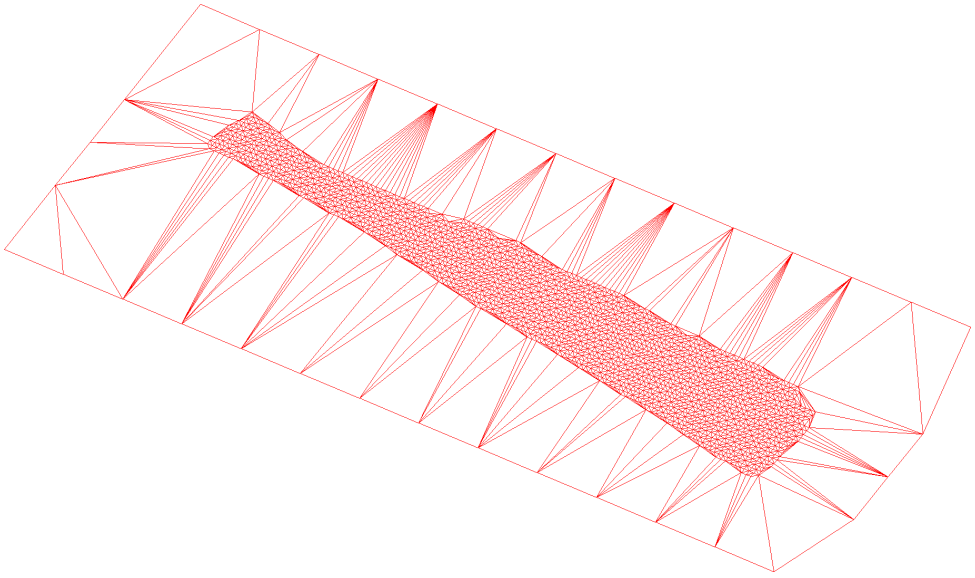


Abbildung 3.21: Delaunay-Triangulation für die Verschiebungspunkte innerhalb und außerhalb der Bathymetrie mit 20 m Abstand

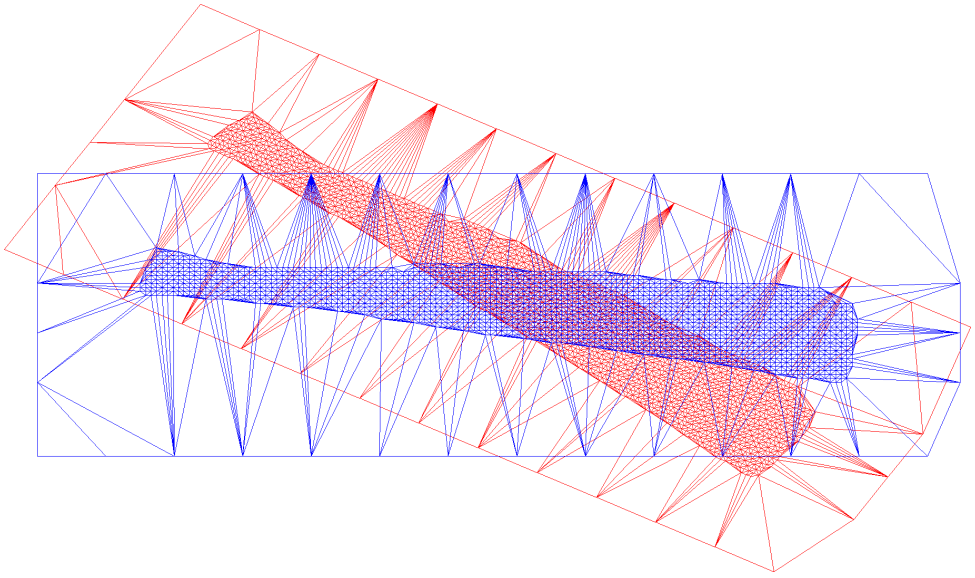


Abbildung 3.22: Delaunay-Triangulation für die Verschiebungspunkte in der Bathymetrie mit 20 m Abstand (rot) und die global rotierte Triangulation der Verschiebungspunkte (blau)

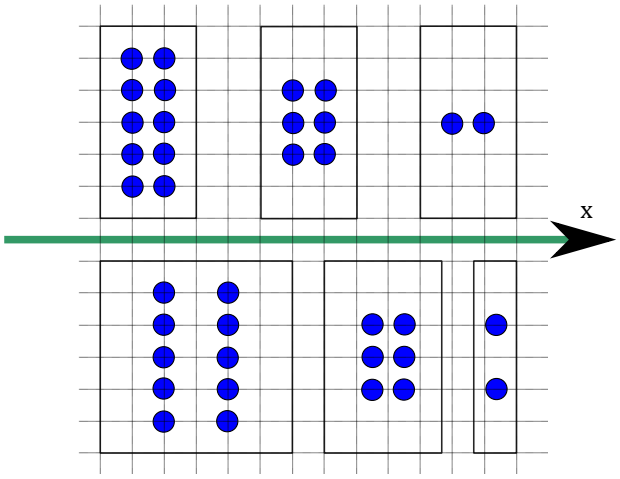


Abbildung 3.23: Beispielhafte Verteilung der x-Werte. Obere Reihe: die gleich großen Blöcke enthalten 10, 6 und 2 Punkte. Untere Reihe: Die x-Wert-Verteilung erzeugt Blockgrößen in den Verhältnissen $\frac{5}{9}$, $\frac{3}{9}$ und $\frac{1}{9}$.

2. Für jeden Block i wird nun die Anzahl der Verschiebungspunkte $\mathbf{n_i}$ bestimmt, die innerhalb des Gebiets der Bathymetrie liegen. Die Gesamtzahl der Verschiebungspunkte ist $N := \sum n_i$.

3. Für jeden Block i wird seine neue Breite berechnet als $l_i := L \cdot \frac{n_i}{N}$.

4. Die Blöcke werden entsprechend ihrer neuen Breite skaliert, d. h. es werden die x-Komponenten der Verschiebungspunkte innerhalb eines Blocks als Ganzes verschoben.

Die Punkte werden somit blockweise parallel zur x-Achse verschoben. Dies führt somit zu einer Streckung von dicht bzw. einer Stauchung von dünn besetzten Gebieten und insgesamt verteilen sich die Punkte gleichmäßiger innerhalb des verfügbaren Gebiets. Eine vereinfachte Darstellung findet sich in Abbildung 3.23, und die Anwendung am Beispiel zeigt Abbildung 3.24.

y-Verschiebung

Nachfolgend zur x-Verschiebung werden die y-Verschiebungen der Verschiebungspunkte nun separat betrachtet. Die Verschiebungspunkte liegen auf einem regelmäßigen Gitter, daher gibt es stets Verschiebungspunkte mit identischen x-Koordinaten.
Die y-Verschiebung ergibt sich nun nach folgendem Ablauf:

1. Bestimme aus allen Verschiebungspunkten den global minimalen (Y^-), maximalen (Y^+) und durchschnittlichen (Y^*) lotrechten y-Abstand zur Hauptachse.

2. Gruppiere die Verschiebungspunkte nach ihren identischen x-Koordinaten.

3. Bestimme für jede verfügbare x-Koordinate die Verschiebungspunkte, die innerhalb eines wählbaren Fensters $[x - w, x + w]$ liegen.

4. Bestimme innerhalb des Fensters die minimalen (y^-), maximalen (y^+) und mittleren (y^*) y-Werte.

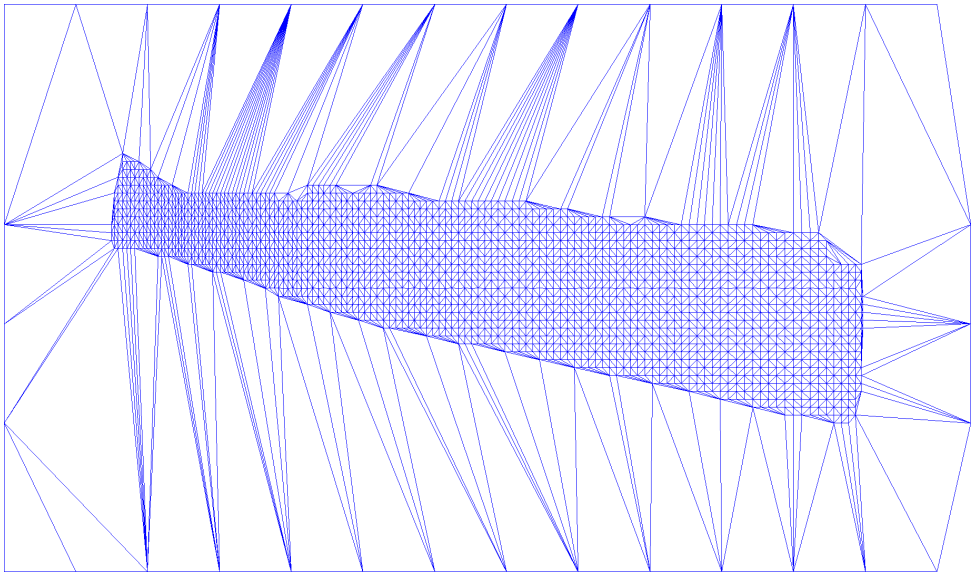


Abbildung 3.24: Neue x-Werte für die Triangulation des Zielgebiets

5. Die neue y-Komponente $\hat{y}$ jedes Verschiebungspunkts ergibt sich anhand des vorherigen y-Werts folgendermaßen:
   - falls $y < y^*$, dann: $\hat{y} := Y^- \cdot (1 - \frac{y-y^-}{y^*-y^-}) + Y^* \cdot \frac{y-y^-}{y^*-y^-}$
   - falls $y > y^*$, dann: $\hat{y} := Y^* \cdot (1 - \frac{y-y^*}{y^+-y^*}) + Y^+ \cdot \frac{y-y^*}{y^+-y^*}$
   - sonst $\hat{y} := Y^*$.

Im Wesentlichen ergibt sich eine gleichmäßige Verteilung der Verschiebungspunkte mit identischer x-Koordinate entlang der y-Ausdehnung der Bathymetrie. Das prinzipielle Schema ist in Abbildung 3.25 dargestellt. Die y-Verschiebung der Verschiebungspunkte im Beispiel mit einer Fenstergröße von 50 m ist abgebildet in Abbildung 3.26.

Kombiniertes Warping

Die Verknüpfung der Rotation mit den lokalen Verschiebungen ergibt eine Menge an Verschiebungsvektoren für das Warping. Jedem Punkt wird genau ein Vektor zugewiesen. Einzelne Verschiebungsvektoren ergeben sich aus der bivariaten Interpolation des Verschiebungsfelds mit Methoden der Scattered-Data-Interpolation. An dieser Stelle wird eine lineare, baryzentrische Dreiecksinterpolation gewählt. [32] empfiehlt hierfür die bikubische Clough-Tocher-Interpolation, da diese innerhalb der Dreiecke eine sanftere (da kubische) Interpolation vorgibt. Die hierfür notwendige innere Zerlegung der Dreiecke und die Berechnung der Zwischenpunkte ist für die oben beschriebene Warping-Methode allerdings mit zusätzlichem Aufwand verbunden. Da visuelle Aspekte keine Rolle spielen (die transformierte Bathymetrie wird ausschließlich algorithmisch betrachtet), ist der Effekt der unstetigen Interpolation zu vernachlässigen.

Zusätzlich ist zu beachten, dass die Warping-Transformation i. A. keine invertierbare Transformation darstellt. Unter Umständen wird die Bathymetrie derart verzerrt, dass sich schneidende oder kreuzende Teilbereiche entstehen. Mit einer geeigneten Anpassung der Parameter für die Blockgröße bzw. der Fensterbreite für die x- und y-Verschiebung entfällt dies. Diese sind derart zu justieren, dass keine sich gegenseitig kreuzenden Verschiebungsvektoren entstehen.

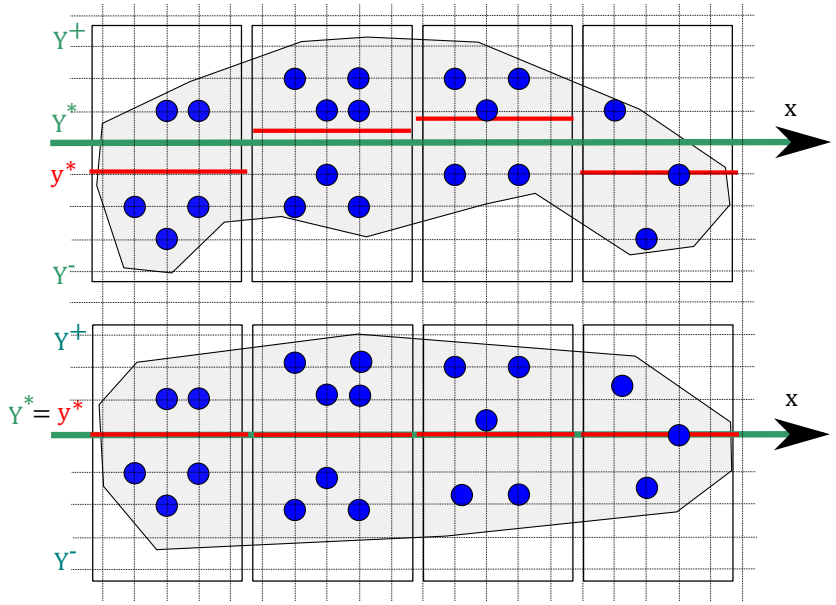


Abbildung 3.25: Darstellung der alten (oben) und der neuen (unten) Verteilung der y-Werte in jedem Fenster

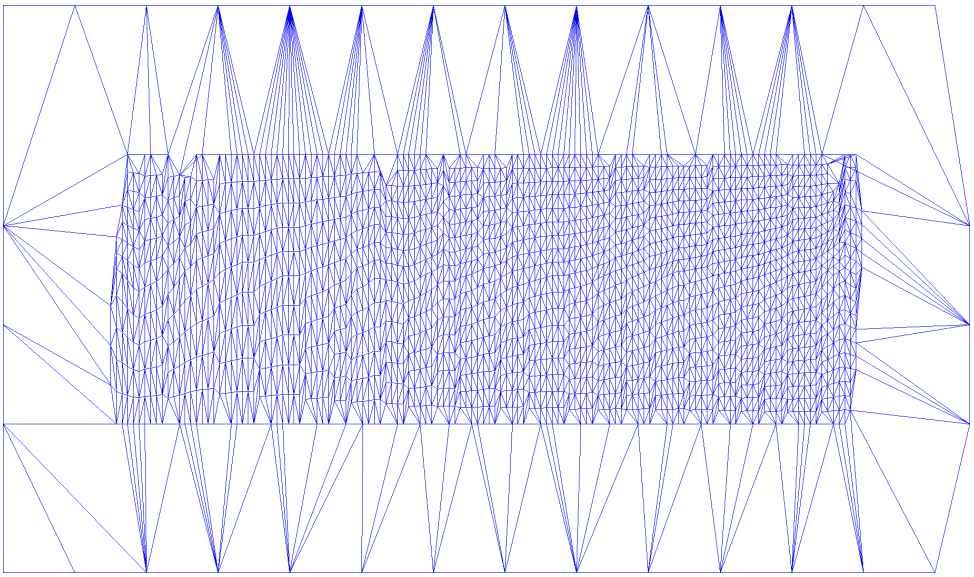


Abbildung 3.26: Neue y-Werte für die Triangulation des Zielgebiets

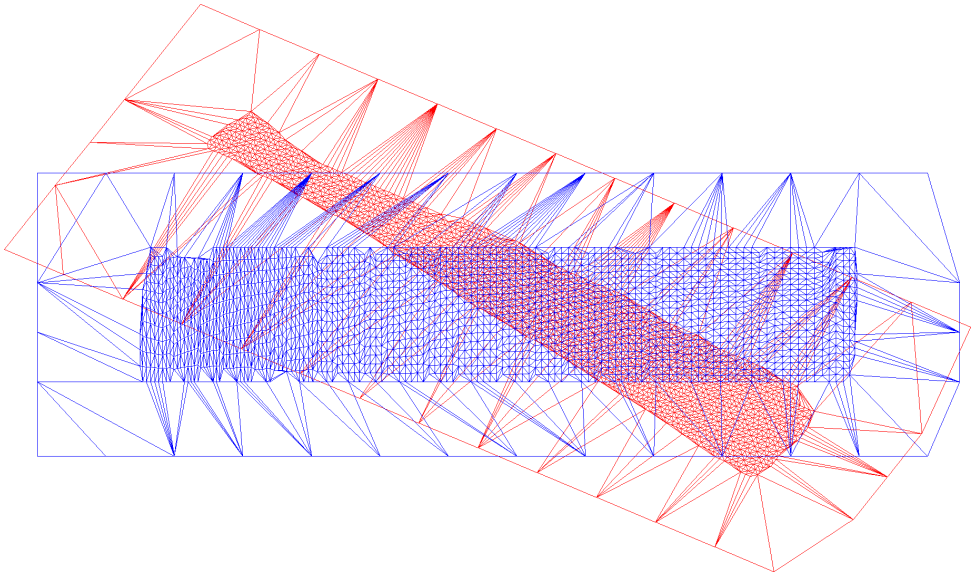


Abbildung 3.27: Darstellung der kombinierten x- und y-Transformation

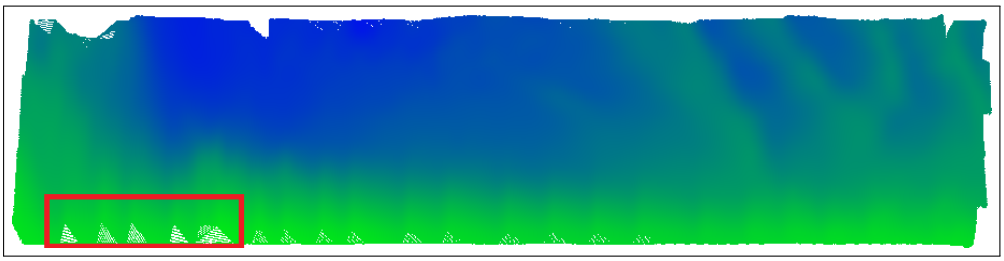


Abbildung 3.28: Vorwärtstransformation der Bathymetrie

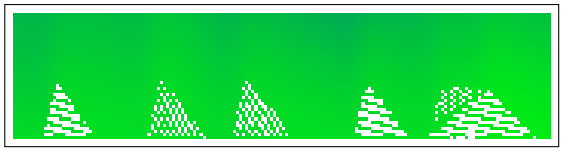


Abbildung 3.29: Detail der Vorwärtstransformation

Abbildung 3.27 zeigt die vollständig transformierte Triangulation. Die nördlichen Gebiete der ursprünglichen Bathymetrie stauchen sich sichtbar in der negativen x-Richtung und sind in positive y-Richtung ausgedehnt. Insgesamt deckt die transformierte Triangulation das rechteckige Gebiet der Bathymetrie beinahe vollständig ab. Das neue Dreiecksnetz verletzt u. U. allerdings die Umkreisbedingung der Delaunay-Triangulation.

Die Vorwärtstransformation bildet nun die (verrasterte) Bathymetrie anhand des Verschiebungsfelds auf ein neues Raster ab. Das Ergebnis zeigt Abbildung 3.28. Deutlich zeigt sich hier eine Lückenbildung der Geometrie an den Rändern (Detail der Lückenbildung in Abbildung 3.29), einen Lösungsansatz hierfür bietet das erwähnte Gauß-Splatting (Abbildung 3.30). Die neue Geometrie verkörpert allerdings weiterhin keine kontinuierliche, d. h. das Raster vollständig ausfüllende, Bathymetrie.

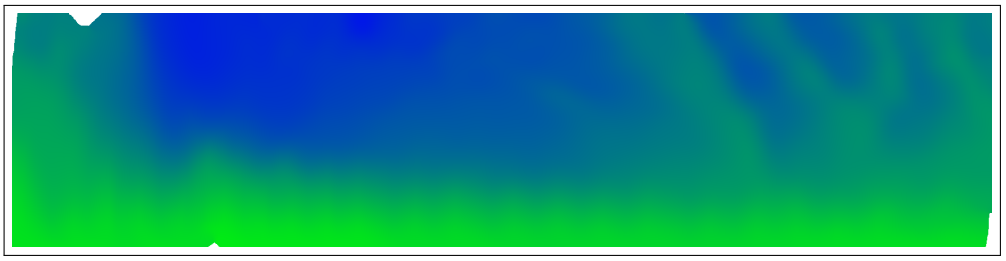


Abbildung 3.30: Vorwärtstransformation der Bathymetrie mit Gauß-Splatting. Teile des Gebiets bleiben weiterhin unbekannt.

3.3.3 Dilatation

Unbekannte Tiefenwerte am Rand oder außerhalb einer vermessenen Bathymetrie werden durch eine Extrapolation der bekannten Werte bestimmt. Die Dilatation ist ein Verfahren der Bildverarbeitung zum Vergrößern von Gebieten (z. B. Linienverdickung). Anstelle einer 0-1-Maskierung wird nun jeder unbekannte Tiefenwert aus den umliegenden bekannten Werten extrapoliert. Algorithmus 2 beschreibt das Verfahren im Detail.

begin
  while *es gibt einen Rasterpunkt* ***p*** *ohne Tiefenwert* **do**
    if ***p*** *hat mindestens einen Nachbarn* **then**
      – Bestimme sämtliche umliegenden Rasterpunkte
      – Setze den neuen Tiefenwert als gewichtete Extrapolation der umliegenden Rasterpunkte
    end
  end
end

Algorithmus 2 : Extrapolation von Tiefenwerten

Die Nachbarschaft kann wahlweise eine 4er- oder eine 8er-Beziehung (d. h. Anzahl der betrachteten Richtungen) sein. Beispiele hierfür sind Abbildungen 3.31 und 3.32. Sobald sämtliche Rasterpunkte im Messgebiet einen Tiefenwert besitzen, endet das Verfahren. Abbildung 3.33 zeigt das Ergebnis am verwendeten Beispieldatensatz.

3.3.4 Approximation und Flächenrückführung

W Mit der direkten Reduktion wird ein Approximationsschema für die Flächenrückführung eingeführt, das identisch mit diesem Schritt ist (Abschnitt 3.2.2). Anstelle der bathymetrischen Vermessungsdaten wird nun abweichend die transformierte Bathymetrie des Vorgängerschritts verwendet. Als Ergebnis entsteht ein T-Netz, das eine optimierte geometrische Fläche für die transformierte Bathymetrie repräsentiert.

Die Abbildungen 3.34 bis 3.37 stellen die progressive Netzverfeinerung für den indirekten Prozess dar (jeweils als Netz, kubische T-Spline-Fläche und Fehlerklassen).

3.3.5 Rücktransformation und Export

Das Approximationsschema erzeugt ein T-Netz für die T-Spline-Fläche der transformierten Bathymetrie. Als abschließender Schritt ist die kombinierte Transformation

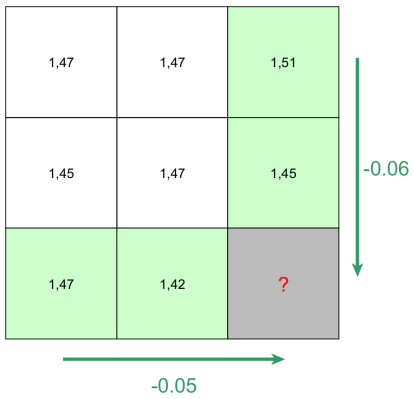


Abbildung 3.31: Extrapolation bei 4er-Nachbarschaft: (?) ergibt sich als $\frac{1}{2} \cdot (1,45 - 0,06) + \frac{1}{2} \cdot (1,42 - 0,05) = 1,38$.

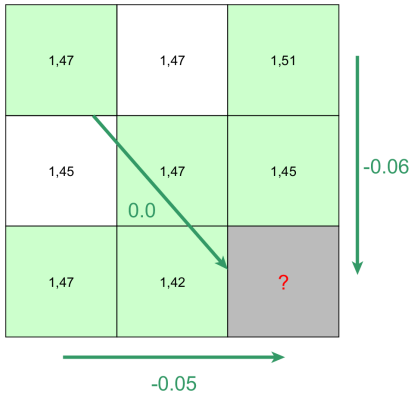


Abbildung 3.32: Extrapolation bei 8er-Nachbarschaft: (?) ergibt sich als $\frac{1}{3} \cdot (1,45 - 0,06) + \frac{1}{3} \cdot (1,42 - 0,05) + \frac{1}{3} \cdot (1,47 - 0,0) = 1,41.$

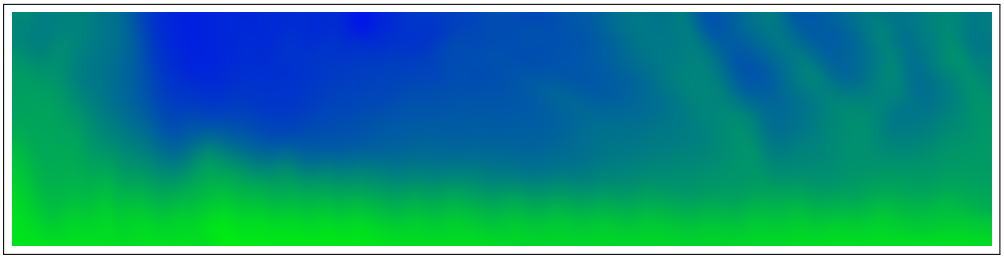


Abbildung 3.33: Transformation und Dilatation der Bathymetrie aus Abbildung 3.30

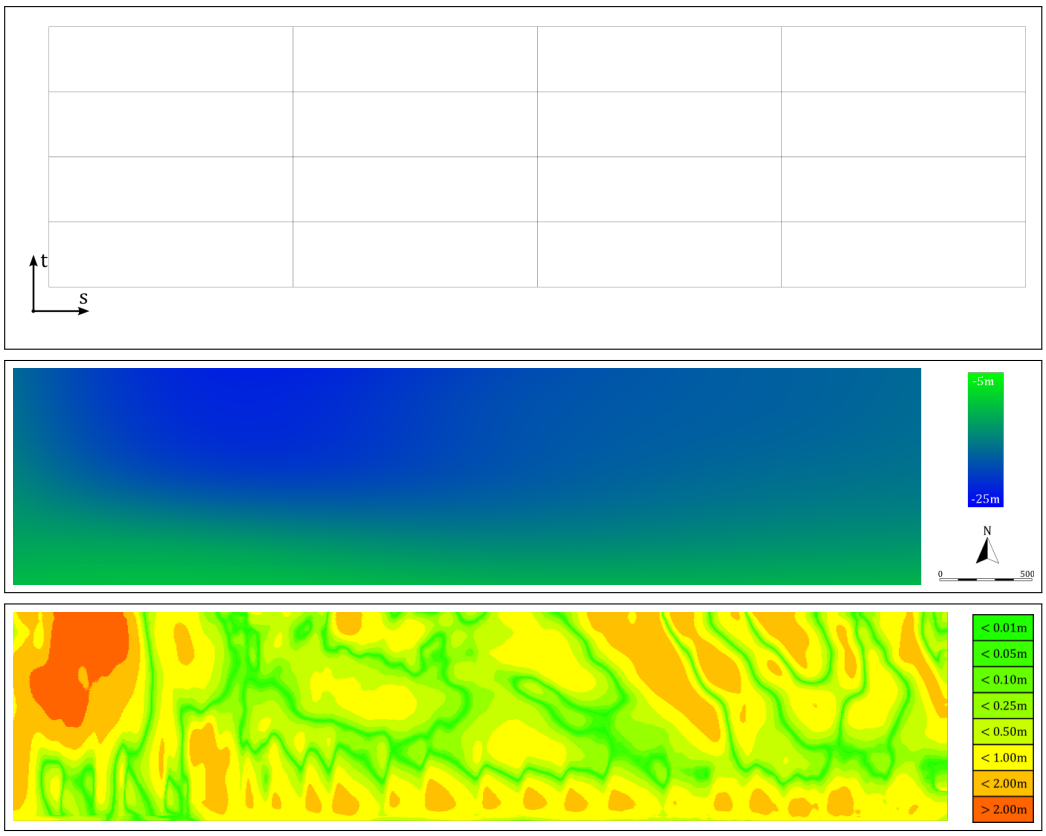


Abbildung 3.34: Initiales T-Netz, Fläche und Klassifikation der Differenzfehler der transformierten Bathymetrie

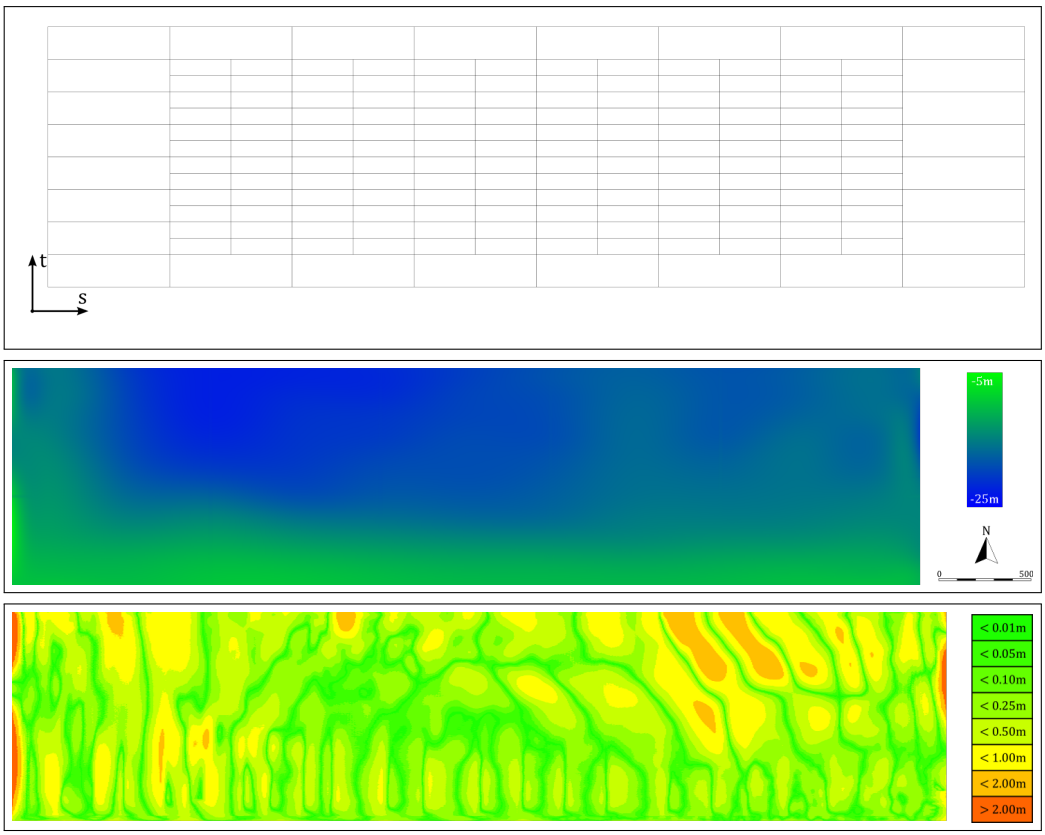


Abbildung 3.35: T-Netz, Fläche und Klassifikation der Differenzfehler der transformierten Bathymetrie (2. Iteration)

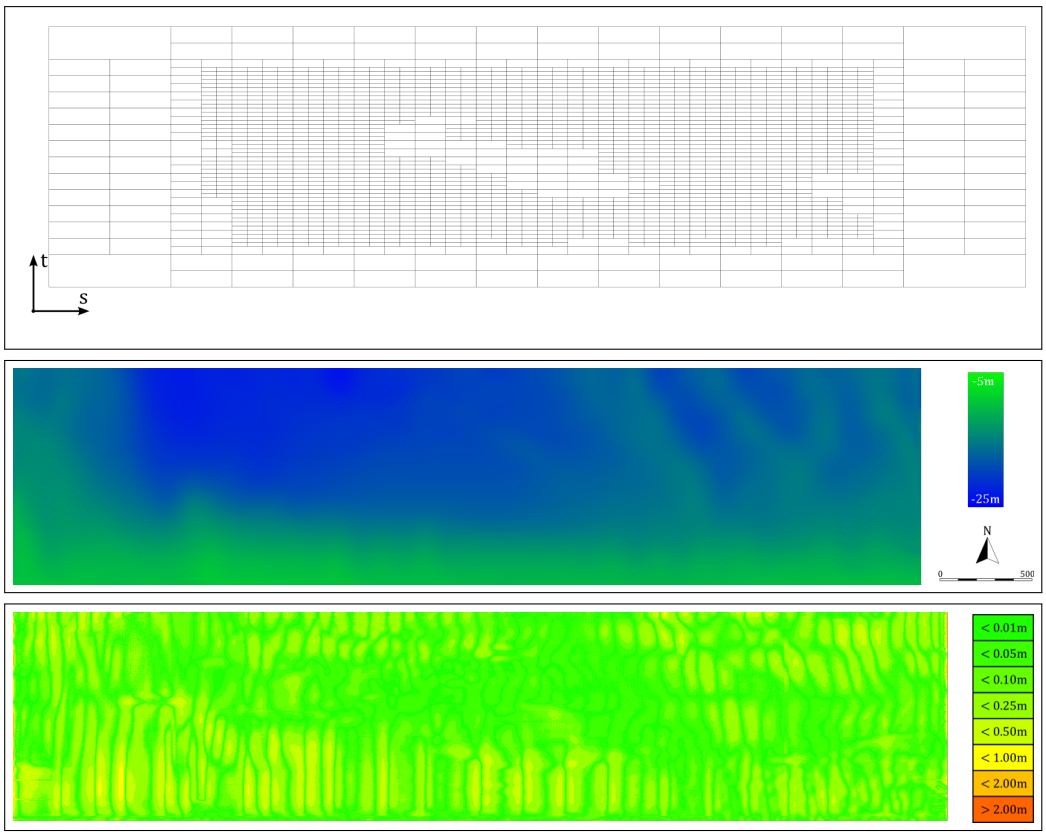


Abbildung 3.36: T-Netz, Fläche und Klassifikation der Differenzfehler der transformierten Bathymetrie (4. Iteration)

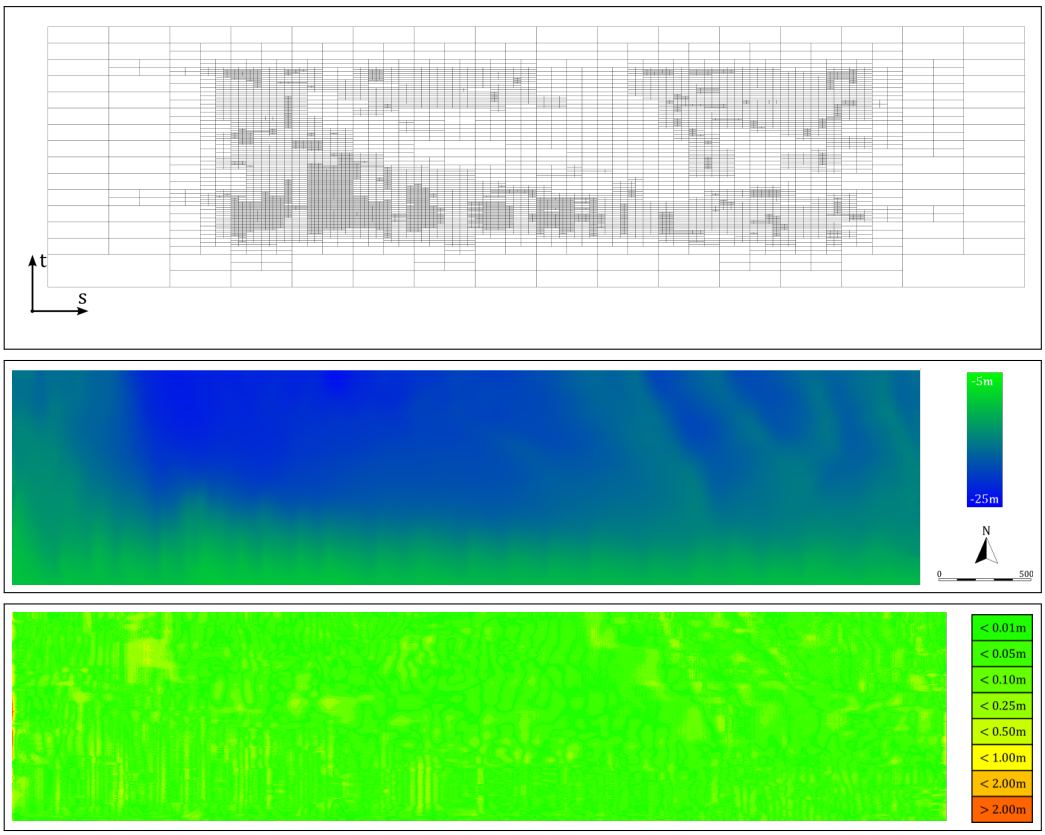


Abbildung 3.37: T-Netz, Fläche und Klassifikation der Differenzfehler der transformierten Bathymetrie (6. Iteration)

zu invertieren, so dass letztlich eine angenäherte T-Spline-Fläche für die ursprüngliche Bathymetrie entsteht.

Ein Kontrollpunkt im T-Netz besteht aus einer Parameter- und einer Geometriekoordinate. Das Approximationsschema erzeugt ein T-Netz mit einer starken Kopplung zwischen diesen Tupeln, d. h. sie liegen zunächst räumlich dicht beieinander. Die Kontrollpunkte des T-Netzes werden nun einzeln zurück transformiert, d. h. jeder Geometriepunkt wird anhand des invertierten Verschiebungsfelds auf das Gebiet der ursprünglichen Bathymetrie abgebildet. Die Ergebnisflächen und die Klassifikation der Differenzfehler für die Rücktransformation sind in den Abbildungen 3.38 bis 3.40 sichtbar. Die T-Netze entsprechen dabei weiterhin den vorher hergeleiteten (Abbildungen 3.36 bis 3.37), sie sind prinzipiell nicht deckungsgleich mit den neuen Flächen für die reduzierten Bathymetrien. Einen Vergleich der Kennzahlen bietet Abbildung 3.41 und die Verteilung der Differenzfehler Abbildung 3.42.

3.3.6 Vergleich zwischen direktem und indirektem Vorgehehen

Es zeigt sich, dass qualitativ und visuell vergleichbare Flächenmodelle vorzugsweise mit der indirekten Reduktion erzeugt werden können. Hier ist es bei deutlich geringerer Netzgröße möglich, eine T-Spline-Fläche für eine Bathymetrie mit ähnlicher Fehlerklassifikation zu erstellen. Bereits vier Iterationen der indirekten Reduktion erzeugen ein visuell gutes Resultat (anstelle von sechs Iterationen bei der direkten Reduktion). Vergleicht man sowohl den Mittelwert mit der positiven Standardabweichung[9] als auch den Median der Differenzfehler, zeigt sich, dass die indirekte Reduktion T-Netze mit deutlich besserer Charakteristik nach der gleichen Anzahl Iterationen erzeugt. Diese Erkenntnisse werden somit direkt auf die Reduktion der vollständigen Beispielbathymetrie in Kapitel 7 übertragen.

3.4 Zwischenstand

Der beschriebene Prozess strukturiert die Reduktion von bathymetrischen Geländemodellen und erzeugt T-Spline-Flächen mit zunehmender Approximationsgüte. Eine Erweiterung der direkten Reduktion um eine vorherige und abschließende Transformation der Bathymetrie führt zu deutlich ausgeglicheneren T-Netzen. Ein durchgängiges Beispiel mittels linearer Optimierung, basierend auf der Methode der kleinsten Quadrate, demonstriert die allgemeine Machbarkeit.

Das nachfolgende Kapitel geht detailliert auf das modulare Approximationsschema, die lineare Optimierung basierend auf der Methode der kleinsten Quadrate und das Dragging-Verfahren ein. Ein visueller und ein quantitativer Vergleich der beiden Verfahren erfolgt anhand eines realen Beispiels in Kapitel 7.

Prinzipiell führen GPGPU-Programme häufig wiederkehrende Berechnungsschritte effizient durch. Die Methoden der Bildverarbeitung sind hierfür sehr gut geeignet, nicht zuletzt aufgrund der technisch-historischen Nähe. Wesentliche Aufgaben der Reduktion werden durch GPGPU-Entwicklungen optimiert. Kapitel 6 erläutert die hierdurch umgesetzten Operationen.

[9] Die Standardabweichung σ einer Messgröße drückt aus, dass ca. 68% aller Messwerte im Intervall $\mu \pm \sigma$ um den Mittelwert μ liegen. Hier wird nun die positive Standardabweichung $\mu + \sigma$ gewählt, da Fehler unterhalb μ in kleinere Fehlerklassen fallen.

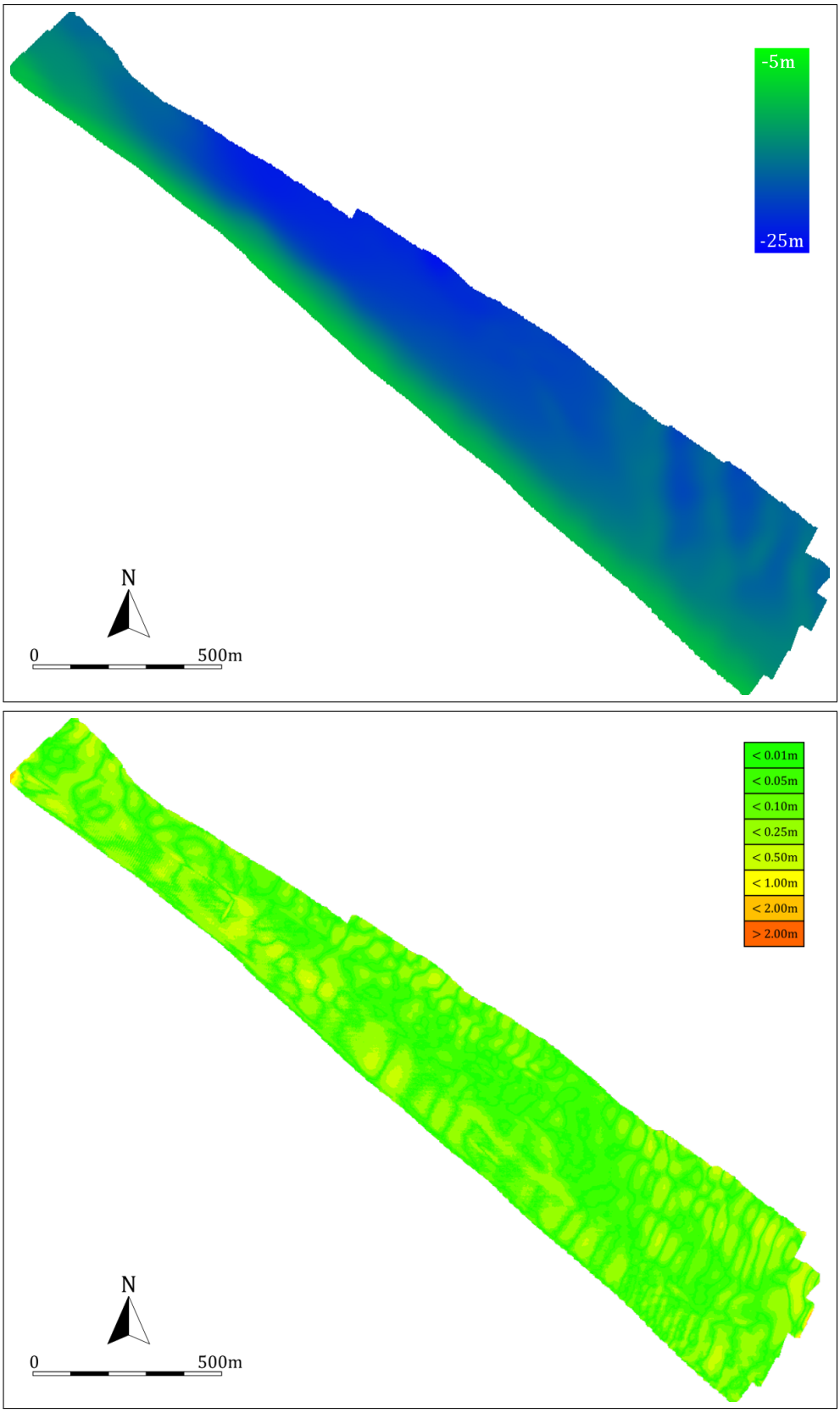


Abbildung 3.38: Finales T-Netz nach der 4. Iteration, T-Spline-Fläche und Klassifikation der Differenzfehler in der indirekten Reduktion

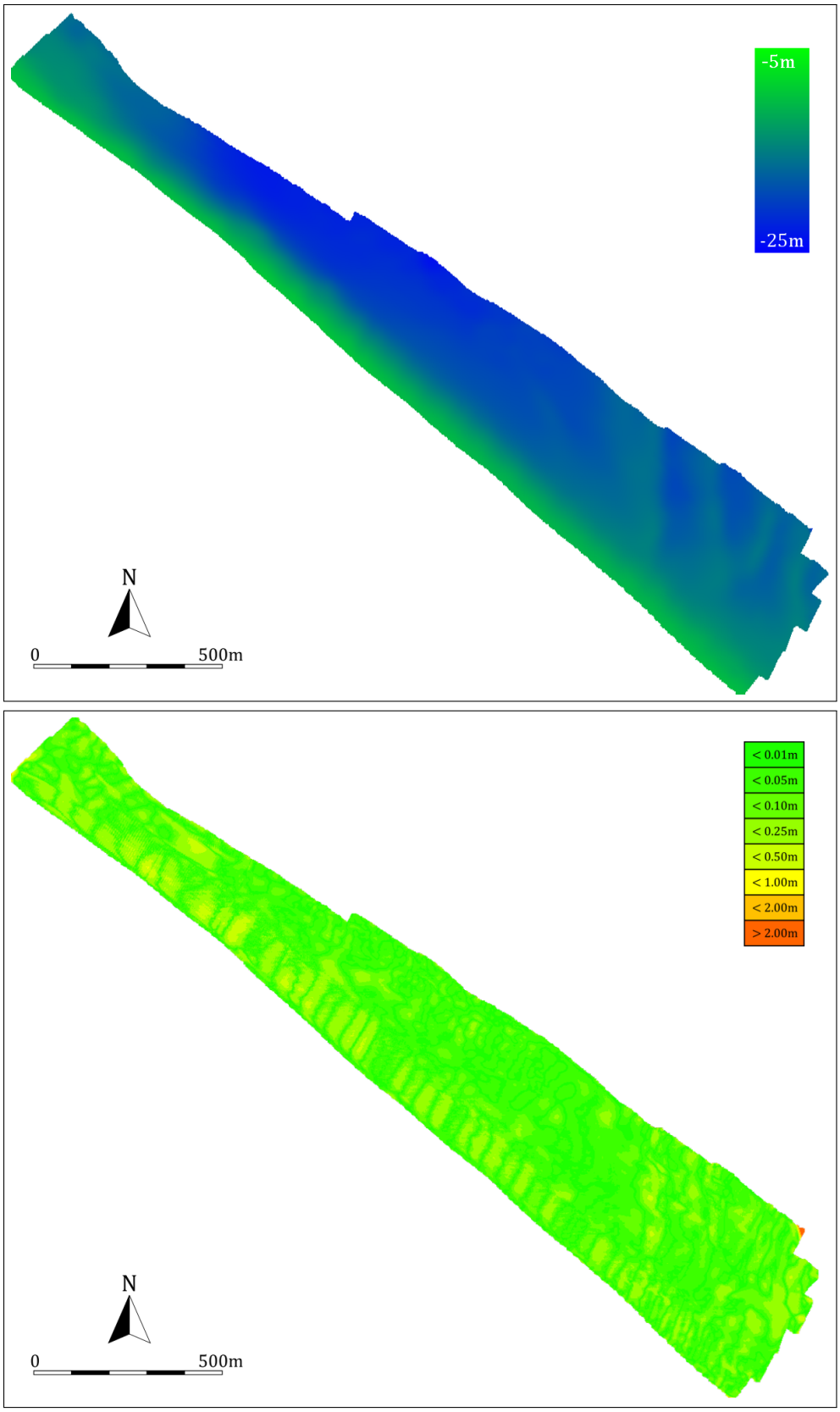


Abbildung 3.39: Finales T-Netz nach der 5. Iteration, T-Spline-Fläche und Klassifikation der Differenzfehler in der indirekten Reduktion

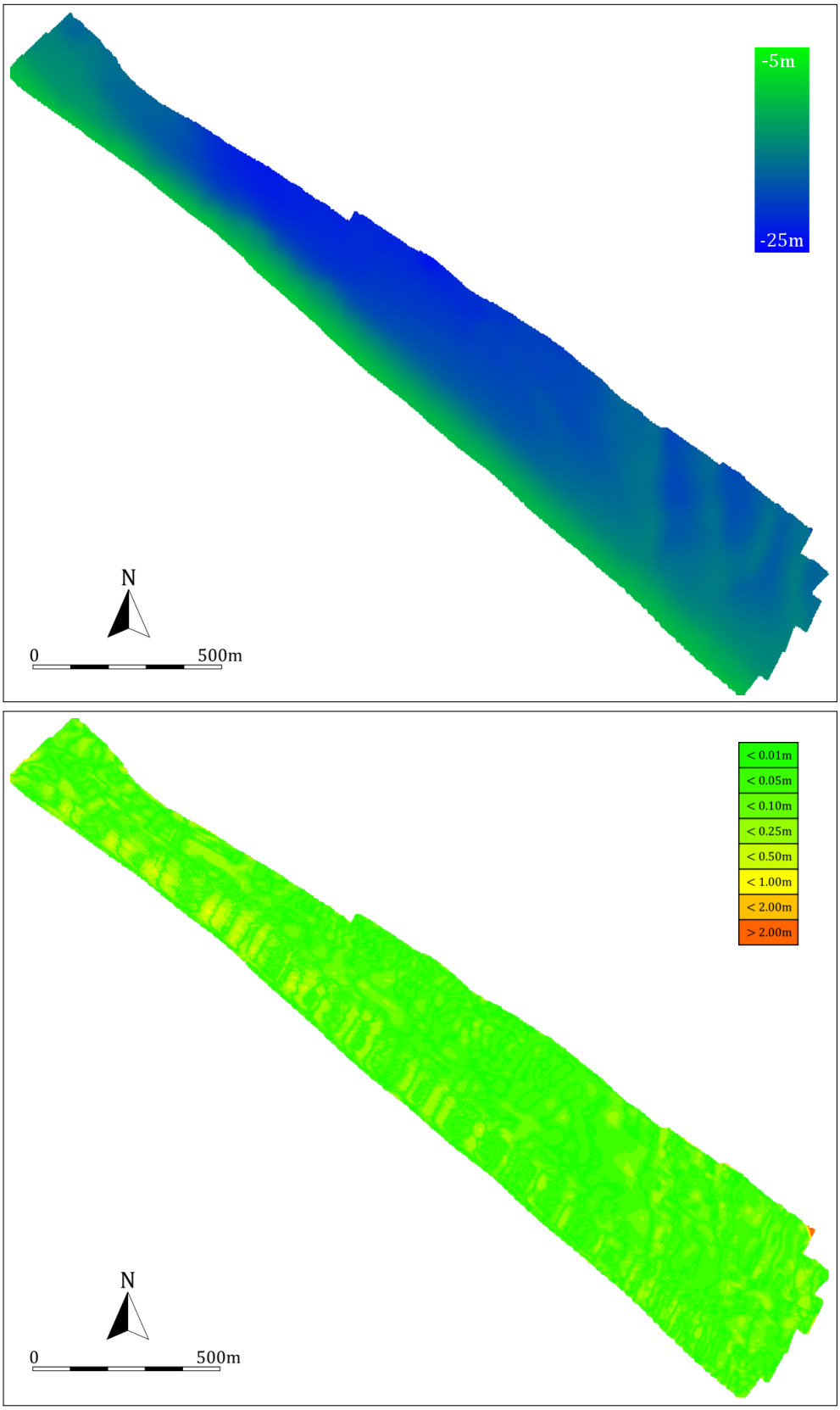


Abbildung 3.40: Finales T-Netz nach der 6. Iteration, T-Spline-Fläche und Klassifikation der Differenzfehler in der indirekten Reduktion

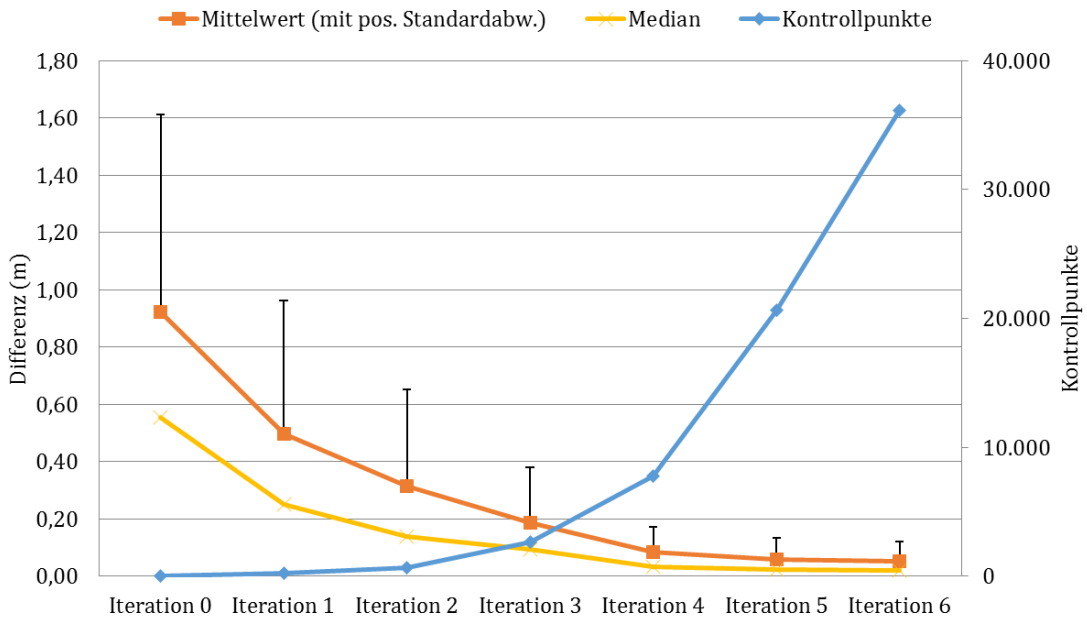


Abbildung 3.41: Vergleich der Qualität der indirekten Optimierung (bezogen auf Mittelwert, Standardabweichung und Median der Differenzfehler) bei zunehmender Anzahl an Kontrollpunkten

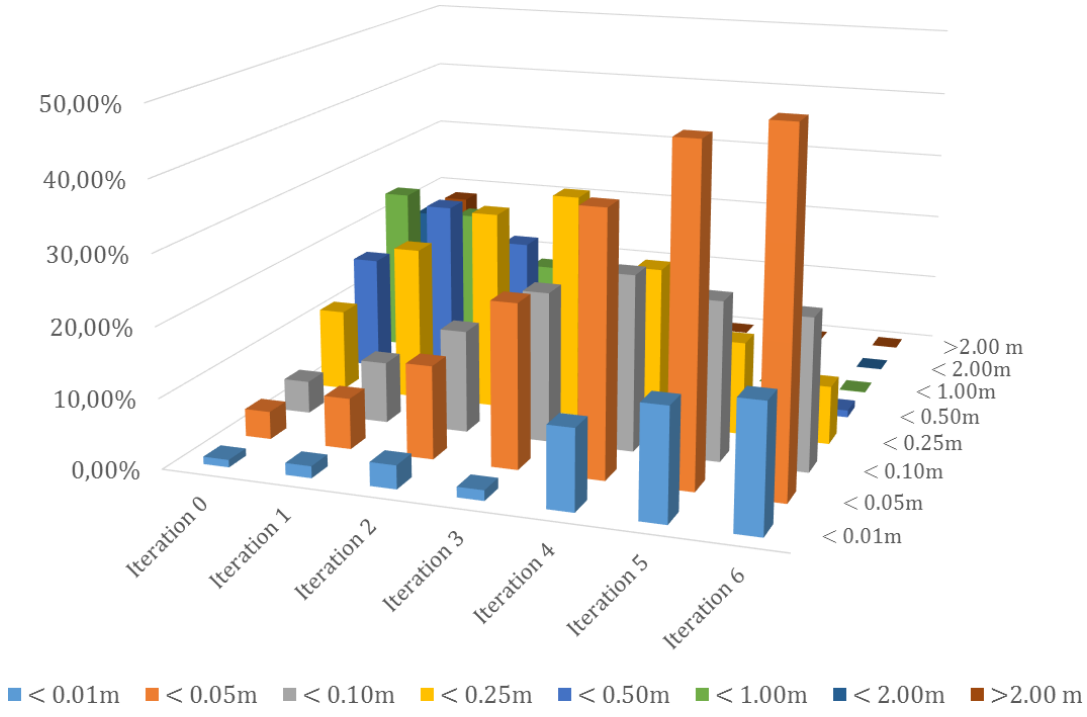


Abbildung 3.42: Histogramm der Fehlerklassen der indirekten Optimierung

4 Approximation und Flächenrückführung

Die Flächenrückführung mit NURBS- und T-Spline-Flächen besitzt Gemeinsamkeiten, führt allerdings auch zu unterschiedlichen Herangehensweisen. Das in diesem Kapitel vorgestellte Approximationsschema für T-Netze lehnt sich daher an bekannte, auf NURBS-Flächen basierende Verfahren an. Darauf aufbauend übernehmen lineare Lösungsmodelle die bathymetrische Flächenrückführung mit T-Spline-Flächen und werden auf ihre Eignung hin untersucht. Bisherige Forschungen beschränken sich auf die vereinfachte Form der Flächenrückführung mit Standard-T-Spline-Flächen. Die Berücksichtigung von Gewichten und Unstetigkeiten am Rand in den Knotenwerten bei Nicht-Standard-T-Spline-Flächen bleibt unbeachtet und wird in dieser Arbeit aufgeholt. Ein iterativ arbeitendes Dragging-Verfahren, das erstmalig für die Flächenrückführung von T-Spline-Flächen beschrieben wird, ergänzt das Approximationsschema um eine modulare Alternative.

4.1 Approximation von T-Spline-Flächen

Im Folgenden wird ein Approximationsschema speziell für die Erzeugung von T-Netzen für T-Spline-Flächen beschrieben. Im Unterschied zur Approximation von rationalen Spline-Flächen (Abschnitt 2.3.1) wird hier der Fokus auf die lokale Modellierung gelegt, die deutlich mehr Freiheitsgrade bietet.

Die Approximation basiert auf einer Menge an Messpunkten der Referenzfläche, einem Ansatzgrad K für die T-Spline-Fläche und einem maximal tolerierten Fehler $e_{\max}$ für die Abweichung. Dies führt zu dem Approximationsschema (Algorithmus 3) basierend auf einer FIFO-Datenstruktur[1].

4.1.1 Initiales T-Netz als Eingabe

Das Approximationsschema beginnt mit der Vorgabe eines T-Netzes mit einer minimal benötigten Anzahl an Kontrollpunkten für die Berechnung einer T-Spline-Fläche. Dieses kann eine beliebige Form annehmen, in der Praxis ist ein rechteckiges Kontrollpunktgitter mit 4×4 oder 5×5 Zellen bereits ausreichend. Das T-Netz deckt die zu modellierende Geometrie vollständig ab. Um eine ausreichend gute Randerfassung zu gewährleisten, wird ein wählbarer Überhang modelliert, in Abbildung 4.1 sind dies 50% der Ausdehnung in jede Richtung. Die durch das Approximationsschema ausgeführten Zellverfeinerungen finden innerhalb des vermessenen Gebiets der Bathymetrie statt. Weitere Verfeinerungen resultieren aus der nachträglichen Korrektur des T-Netzes (s. u.). Somit ist der zusätzliche Bedarf an Kontrollpunkten im T-Netz zu vernachlässigen.

4.1.2 Netzverfeinerung

Nach jedem Schritt der Approximation liegt ein T-Netz vor, in dem für jeden Kontrollpunkt die lokalen Knotenwerte und die Tiefenwerte bekannt sind. Der Fehler $e_{\max}$ stellt

[1] **F**irst **I**n - **F**irst **O**ut — eine abstrakte Datenstruktur, bei der das älteste eingefügte Elemente als erstes ausgelesen wird.

Eingabe: Ein T-Netz mit minimaler Anzahl an Kontrollpunkten
– Initiiere die leere FIFO-Datenstruktur $C := \emptyset$ für die Zellen im T-Netz
– Bestimme die Menge B der Gebiete auf der T-Spline-Fläche mit punktuellem Differenzfehler $\geq e_{\max}$
– Füge die Zellen des T-Netzes in C ein, deren Kontrollpunkte die Gebiete B der T-Spline-Fläche direkt erzeugen
begin
  while $C \neq \emptyset$ **do**
    – Verfeinere die T-Spline-Flächen in den Zellen $c \in C$
    – Erzeuge eine T-Spline-Fläche vom Grad K über dem T-Netz
    – Führe die Optimierung der Tiefenwerte im T-Netz durch
    – Aktualisiere die Menge C mit den neuen Gebieten B' mit einem Differenzfehler $\geq e_{\max}$
  end
end
Ausgabe: Ein optimiertes T-Netz zu einer T-Spline-Fläche mit minimiertem Differenzfehler

Algorithmus 3 : Approximationsschema für T-Netze

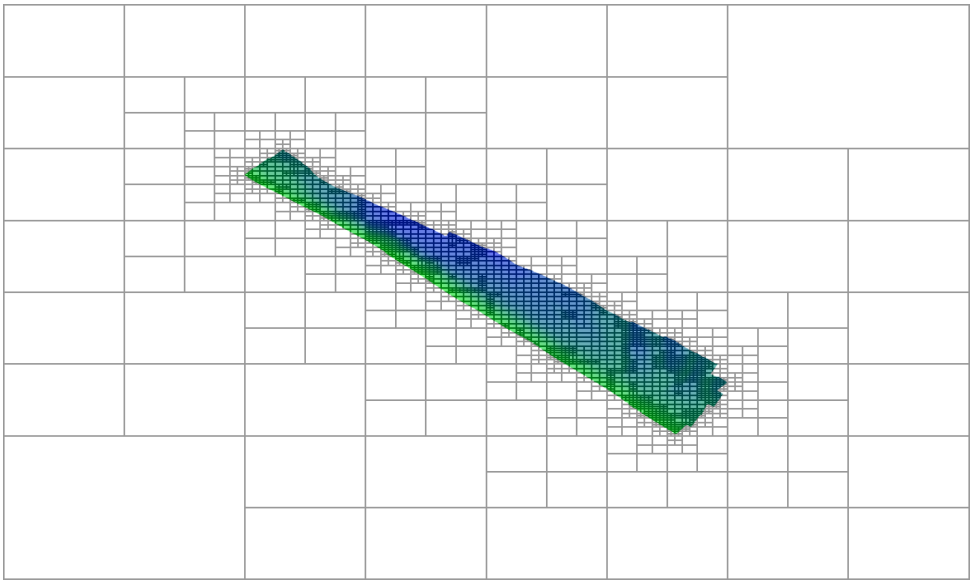


Abbildung 4.1: Bathymetrie mit überhängendem T-Netz

die maximale Abweichung der Tiefe eines beliebigen Punkts auf der T-Spline-Fläche im Vergleich zu seinem korrespondierenden interpoliertem Punkt in den Rasterdaten der Bathymetrie dar. Voraussetzung hierfür ist die Existenz und Berechenbarkeit dieser beiden Punkte.

Jede Zelle des T-Netzes wird durch ein Gitter, bestehend aus 3×3 Punkten, überlagert. Insgesamt ergibt dies neun Punkte (vier Eckpunkte, vier Randpunkte und ein zentraler Punkt). Eine Zelle wird verfeinert, wenn beide Bedingungen zutreffen:

- Mindestens ein Punkt der Zelle liegt innerhalb der Bathymetrie.
- Mindestens ein Punkt der Zelle besitzt einen Differenzfehler $\geq e_{\max}$.

Liegen alle Punkte der Zelle vollständig außerhalb der Bathymetrie, oder ist die punktuelle Abweichung zu gering, so ist keine Verfeinerung nötig.

Ist die Menge der zu verfeinernden Zellen bekannt, werden diese lokal verfeinert, d. h. sie werden in vier Unterzellen unterteilt. Das Vorgehen ist prinzipiell in Abschnitt 5.4.4 und konkret für die eingesetzten Rechteck-T-Netze in Abschnitt 5.5.2 beschrieben.

Somit entstehen neue Kontrollpunkte, deren Tiefenwerte und Knotenwerte (und ggf. Gewichte) unbekannt sind. Die Kontrollpunkte der vorherigen Iterationen müssen ebenfalls angepasst werden. Abschnitt 4.3 erläutert die Verfahren hierzu.

Korrektur des T-Netzes

Das Einfügen von zusätzlichen Kontrollpunkten im Rechteck-T-Netz kann zu einer Verletzung der zweiten Regel für T-Spline-Flächen nach [77] führen (Abschnitt 5.2.3). Hierfür sind zusätzliche Unterteilungen der Zellen durchzuführen.

Die lokale Verfeinerung von T-Netzen führt oftmals zu Konstellationen mit einer lokal stark variierenden Anzahl an Kontrollpunkten. Allgemein empfehlen sich daher maximal 1-zu-2-Nachbarschaftsbeziehungen in einem T-Netz aufgrund der Problematik hängender Knoten (Abschnitt 5.3). Lokale Unterteilungen korrigieren dies.

Beide Korrekturregeln werden nach Bedarf abwechselnd durchlaufen, bis die beschriebenen Fälle nicht weiter auftreten.

4.1.3 Optimierung der Geometrie

Ein T-Netz modelliert eine T-Spline-Fläche vom Grad K. Zunächst ist das Berechnen der lokalen Knotenwerte erforderlich (Abschnitt 5.2). Diese sind abhängig von K und werden nach jeder Änderung des T-Netzes erneut bestimmt, da das Einfügen oder Entfernen von Kontrollpunkten wesentlich die vorhandenen Knotenwerte beeinflusst. Daher ist dieser Schritt ein häufiger, zeitkritischer Bestandteil der Approximation. Der hohe Anteil der Parallelität (lokale Knotenwerte sind voneinander unabhängig berechenbar) lässt sich durch GPGPU-Algorithmen effektiv umsetzen (Abschnitt 6.3.2).

Die erzeugten T-Netze müssen unbedingt die Randapproximation der T-Spline-Fläche sichern, d. h. die modellierten Flächen grenzen direkt an den Randpolygonen des T-Netzes an. Nicht-Standard-T-Spline-Flächen ermöglichen dies durch mehrfache (nicht-uniforme) Knotenwerte, erfordern allerdings eine abweichende Flächenrückführung im Vergleich mit dem Vorgehen für Standard-T-Spline-Flächen. Hierfür werden die lineare Optimierung (basierend auf der Methode der kleinsten Quadrate) und das Dragging-Verfahren miteinander verglichen.

Das Ergebnis ist jeweils eine optimierte Geometrie des T-Netzes, die der (indirekt transformierten) Bathymetrie weitgehend entspricht. Konkret erhält jeder Kontrollpunkt im T-Netz eine geometrische Koordinate mit einem Tiefenwert zugewiesen.

4.2 NURBS-Flächenrückführung

Grundlegende Aspekte der Flächenrückführung mit NURBS-Flächen können auf die Flächenrückführung mit T-Spline-Flächen übertragen werden.

4.2.1 Allgemeine Problemstellung

Ein Maß für die Güte der Flächenrückführung ist die Fehlermetrik Formel (2.12). Gesucht werden die Geometriekoordinaten der Kontrollpunkte $\mathbf{C} := (\mathbf{c_0} \dots \mathbf{c_{n-1}})^T$ für ein NURBS-Flächenmodell. Die NURBS-Flächenpunkte sind $\mathbf{F} := (\mathbf{f_0} \dots \mathbf{f_{m-1}})^T$ mit $\mathbf{f_i} := \mathbf{f}(s_i, t_i)$.

Zu jedem Flächenpunkt $\mathbf{f_i}$ wird ein korrespondierender Punkt in der gemessenen Bathymetrie durch bilineare Interpolation der Rasterdaten bestimmt. Ein interpolierter Punkt $\mathbf{r}(s,t) \in \mathbb{E}^3$ an den Parameterwerten $(s,t) \in \mathbb{R}^2$ im Raster R (mit ganzzahligen Koordinaten im Abstand 1) ergibt sich als

$$\begin{aligned}\mathbf{d_i} := \mathbf{r}(s_i, t_i) = (\lceil s \rceil - s)(\lceil t \rceil - t) \cdot \mathbf{r_{11}} + (s - \lfloor s \rfloor)(\lceil t \rceil - t) \cdot \mathbf{r_{21}} \\ + (\lceil s \rceil - s)(t - \lfloor t \rfloor) \cdot \mathbf{r_{12}} + (s - \lfloor s \rfloor)(t - \lfloor t \rfloor) \cdot \mathbf{r_{22}}\end{aligned} \tag{4.1}$$

für die Rasterpunkte $\mathbf{r_{11}} := \mathbf{r}(\lfloor s \rfloor, \lfloor t \rfloor)$, $\mathbf{r_{12}} := \mathbf{r}(\lfloor s \rfloor, \lceil t \rceil)$, $\mathbf{r_{21}} := \mathbf{r}(\lceil s \rceil, \lfloor t \rfloor)$ und $\mathbf{r_{22}} := \mathbf{r}(\lceil s \rceil, \lceil t \rceil)$ mit ganzzahligen Koordinaten.

Diese interpolierten Punkte ergeben nun die Punkte $\mathbf{D} := (\mathbf{d_0} \dots \mathbf{d_{m-1}})^T$, die auf den identischen x-y-Koordinaten der Fläche liegen und somit nur einen abweichenden Tiefenwert z aufweisen. Für diese gilt $\mathbf{d_i} := (x(\mathbf{f_i}), y(\mathbf{f_i}), z(\mathbf{d}(\mathbf{f_i})))$ und $\mathbf{d}(\mathbf{f_i})$ ist der bilinear interpolierte Punkt aus den Rasterdaten der Bathymetrie an der (nicht-ganzzahligen) Koordinate $(x(\mathbf{f_i}), y(\mathbf{f_i}))$.

Im folgenden Optimierungsproblem ist nun der Differenzvektor zwischen den Flächenpunkten $\mathbf{F}$ und den interpolierten Rasterpunkten $\mathbf{D}$ zu minimieren:

$$\begin{aligned}\min ||\mathbf{F} - \mathbf{D}||_2 &:= \min \left|\left| \begin{pmatrix} \mathbf{f_0} - \mathbf{d_0} \\ \vdots \\ \mathbf{f_{m-1}} - \mathbf{d_{m-1}} \end{pmatrix} \right|\right|_2 \\ &= \min \left|\left| \begin{pmatrix} (\sum_l \mathbf{c_l} \cdot \phi_l(s_0, t_0)) - \mathbf{d_0} \\ \vdots \\ (\sum_l \mathbf{c_l} \cdot \phi_l(s_{m-1}, t_{m-1})) - \mathbf{d_{m-1}} \end{pmatrix} \right|\right|_2\end{aligned} \tag{4.2}$$

Die Anzahl der Kontrollpunkte ist n, die der Messpunkte ist m. $\phi(s,t)$ sei eine beliebige (rationale) Basisfunktion am l-ten Kontrollpunkt für eine parametrische Fläche $\mathbf{f}(s,t)$ über den zu optimierenden Geometriekoordinaten $\mathbf{C}$ der Kontrollpunkte[2].

4.2.2 Berechnung von Tiefenwerten

In dieser Arbeit werden die Tiefenwerte an den Kontrollpunkten bestimmt. An die Stelle des Vektors der (dreidimensionalen) Flächenpunkte $\mathbf{F}$ tritt nun der auf die z-Werte reduzierte Vektor $\mathbf{Z_F} := (z(\mathbf{f_0}) \dots z(\mathbf{f_{m-1}}))^T$. Eine Reduzierung der interpolierten

[2] Angelehnt an ***d**ata* und ***c**ontrol points*

Messpunkte $\mathbf{D}$ auf die Tiefenwerte ergibt den Vektor $\mathbf{Z_D} := (z(\mathbf{d_0}) \dots z(\mathbf{d_{m-1}}))^T$. Die neu zu berechnenden Tiefenwerte an den Kontrollpunkten sind weiterhin definiert als $\mathbf{Z_C} := (z(\mathbf{c_0}) \dots z(\mathbf{c_{n-1}}))^T$

Das reduzierte Optimierungsproblem für die Berechnung von Tiefenwerten lautet somit

$$\min ||\mathbf{Z_F} - \mathbf{Z_D}||_2 \,. \tag{4.3}$$

Die Matrix A sei eine beliebige Modellmatrix, d. h. sie enthält die Werte der Basisfunktionen für jeden Kontrollpunkt und gibt somit die affine Kombination der Kontrollpunkte vor. A hat die folgende Struktur:

$$A := \begin{pmatrix} \phi_0(s_0,t_0) & \dots & \phi_{n-1}(s_0,t_0) \\ \vdots & & \vdots \\ \phi_0(s_{m-1},t_{m-1}) & \dots & \phi_{n-1}(s_{m-1},t_{m-1}) \end{pmatrix} \tag{4.4}$$

Insgesamt kann der unbekannte Vektor $\mathbf{Z_C}$ nun aus dem folgenden überbestimmten linearen Gleichungssystem bestimmt werden:

$$A\,\mathbf{Z_C} = \mathbf{Z_D}. \tag{4.5}$$

Nach linksseitiger Ergänzung gilt:

$$A^T A\,\mathbf{Z_C} = A^T\,\mathbf{Z_D}. \tag{4.6}$$

Gleichung (4.6) stellt nun ein lineares Gleichungssystem dar, das mit einem optimierten Lösungsverfahren für die Invertiertung bzw. Zerlegung von dünnbesetzten Matrizen den unbekannten Vektor $\mathbf{Z_C}$ erzeugt.

Flächenrückführung mit B-Spline-Basisfunktionen

Vereinfacht gilt $N_l^K(s_k,t_k) := B_i^K(s_k) \cdot B_j^K(t_k)$ und l sei ein eindimensionaler Index der Kontrollpunkte $\mathbf{p_{i,j}}$ im Gitter. Die Flächenrückführung mit B-Spline-Basisfunktionen (Formel (2.3)) erfolgt über die Methode der kleinsten Quadrate [37].

Sei

$$B := \begin{bmatrix} N_0^K(s_0,t_0) & \dots & N_{n-1}^K(s_0,t_0) \\ \vdots & \ddots & \vdots \\ N_0^K(s_{m-1},t_{m-1}) & \dots & N_{n-1}^K(s_{m-1},t_{m-1}) \end{bmatrix} \tag{4.7}$$

eine $n \times m$-Modellmatrix. $\mathbf{Z_D}$ ist weiterhin der Vektor der Messpunkte. Die unbekannten Tiefenwerte der Kontrollpunkte $\mathbf{Z_C}$ ergeben sich als Lösung des linearen Gleichungssystems

$$\begin{aligned} & B\,\mathbf{Z_C} = \mathbf{Z_D} \\ \Leftrightarrow\ & B^T B\,\mathbf{Z_C} = B^T\,\mathbf{Z_D}. \end{aligned} \tag{4.8}$$

Flächenrückführung mit NURBS-Basisfunktionen

Das Problem verschärft sich mit rationalen Basisfunktionen für NURBS-Flächen deutlich. Die Modellmatrix für NURBS-Basisfunktionen (2.5) sieht wie folgt aus:

$$B := \begin{bmatrix} \frac{1}{f_0} \cdot w_0 \cdot N_0^K(s_0,t_0) & \dots & \frac{1}{f_0} \cdot w_n \cdot N_{n-1}^K(s_0,t_0) \\ \vdots & \ddots & \vdots \\ \frac{1}{f_m} \cdot w_0 \cdot N_0^K(s_{m-1},t_{m-1}) & \dots & \frac{1}{f_{m-1}} \cdot w_{n-1} \cdot N_{n-1}^K(s_{m-1},t_{m-1}) \end{bmatrix} \tag{4.9}$$

mit

$$f_k := \sum_{l=0}^{n-1} w_l \cdot N_l^K(s_k,t_k).$$

Diese Matrix entspricht wesentlich der später eingeführten Modellmatrix (4.29) in der linearen Optimierung für Nicht-Standard-T-Spline-Flächen. Die unbekannten Tiefenwerte der Kontrollpunkte $\mathbf{Z_C}$ sind weiterhin Lösung der Gleichung (4.8), allerdings beinhaltet das Gleichungssystem jetzt die ebenso unbekannten Gewichte w_l. Daher wird eine getrennte Optimierung von Gewichten und Geometrie durchgeführt.

4.2.3 Optimierung der Gewichte bei NURBS-Flächen

In dieser Arbeit wird ein Berechnungsschema von Heidrich für die Flächenrückführung mit NURBS-Basisfunktionen auf den einfachen Fall der Berechnung von Tiefenwerten reduziert [37].

Ausgehend von der NURBS-Basisfunktion (2.5) für parametrische Flächen im dreidimensionalen Raum mit

$$\mathbf{f}(s,t) = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} w_{i,j} \cdot \mathbf{c_{i,j}} \cdot B_i^K(s) \cdot B_j^K(t)}{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} w_{i,j} \cdot B_i^K(s) \cdot B_j^K(t)} \tag{4.10}$$

ergeben sich die interpolierten dreidimensionalen Messpunkte $\mathbf{D} = (\mathbf{d_0} \dots \mathbf{d_{k-1}})$ für $l = 0, \dots, k-1$ als

$$\mathbf{d_l} = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} w_{i,j} \cdot \mathbf{c_{i,j}} \cdot B_i^K(s) \cdot B_j^K(t)}{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} w_{i,j} \cdot B_i^K(s) \cdot B_j^K(t)} \,. \tag{4.11}$$

Dies ist gleich

$$\mathbf{d_l} \cdot \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} w_{i,j} \cdot B_i^K(s) \cdot B_j^K(t) = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} w_{i,j} \cdot \mathbf{c_{i,j}} \cdot B_i^K(s) \cdot B_j^K(t) \tag{4.12}$$

und für die separierten Tiefenwerte $z_l := z(\mathbf{d_l})$ gilt

$$z_l \cdot \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} w_{i,j} \cdot B_i^K(s) \cdot B_j^K(t) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} w_{i,j} \cdot z(\mathbf{c_{i,j}}) \cdot B_i^K(s) \cdot B_j^K(t) \; . \tag{4.13}$$

Diese Gleichung soll in ihrer Matrixform dargestellt werden. Die Tiefenwerte der insgesamt $k+1$ interpolierten Messpunkte $\mathbf{Z_D}$ sind als Diagonalmatrix Z definiert als

$$Z := \begin{pmatrix} \ddots & & 0 \\ & \mathbf{Z_D} & \\ 0 & & \ddots \end{pmatrix} = \begin{pmatrix} z(\mathbf{d_0}) & & 0 \\ & \ddots & \\ 0 & & z(\mathbf{d_k}) \end{pmatrix} \tag{4.14}$$

und weiterhin sei

$$C := \begin{pmatrix} z(\mathbf{c_{0,0}}) \cdot w_{0,0} \\ \vdots \\ z(\mathbf{c_{m,n}}) \cdot w_{m,n} \end{pmatrix} \tag{4.15}$$

Die Matrix B entspricht der oben eingeführten Modellmatrix (4.9) für NURBS-Basisfunktionen. Die Matrixform von Gleichung (4.13) lautet daher

$$ZB\,\mathbf{W} = B\mathbf{C} \tag{4.16}$$

und weiter

$$\begin{bmatrix} B & -ZB \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \mathbf{W} \end{bmatrix} = \mathbf{0}. \tag{4.17}$$

Dieses Gleichungssystem wird linksseitig multipliziert[3]:

$$\begin{aligned} & \begin{bmatrix} B - ZB \end{bmatrix}^T \begin{bmatrix} B - ZB \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \mathbf{W} \end{bmatrix} = \mathbf{0} \\ \Leftrightarrow \quad & \begin{bmatrix} B^T \\ -B^T Z \end{bmatrix} \begin{bmatrix} B - ZB \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \mathbf{W} \end{bmatrix} = \mathbf{0} \\ \Leftrightarrow \quad & \begin{bmatrix} B^T B & -B^T ZB \\ -B^T ZB & -B^T Z^2 B \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \mathbf{W} \end{bmatrix} = \mathbf{0} \end{aligned} \tag{4.18}$$

Im folgenden Schritt wird die obere Gleichung umgeformt zu

$$\begin{bmatrix} B^T B & -B^T ZB \\ 0 & M \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \mathbf{W} \end{bmatrix} = \mathbf{0} \tag{4.19}$$

mit

$$M := B^T Z^2 B - (B^T ZB)(B^T B)^{-1}(B^T ZB).$$

Gleichung (4.19) beschreibt ein Problem, in dem die Gewichte $\mathbf{W}$ in die Modellfunktion eingehen.

[3] Insbesondere gilt $Z^T = Z$ und somit $Z^T Z = Z^2$.

Ein Lösungsansatz von Heidrich et al. besteht nun darin, diese Optimierung in eine zweistufige lineare Optimierung aufzuteilen. Zunächst werden die Gewichte $\mathbf{W}$ optimiert, anschließend erfolgt die Berechnung der geometrischen Koordinaten und speziell die Berechnung der Tiefenwerte an den Kontrollpunkten [37].
Aus der letzten Zeile von (4.19) folgt

$$M\,\mathbf{W} = \mathbf{0}. \tag{4.20}$$

Der Vektor der Gewichtswerte ist folglich die nicht-triviale Lösung des homogenen Gleichungssystems (4.20), d. h. $\mathbf{W} \neq \mathbf{0}$. Die folgenden Lösungsansätze für (4.20) sind aus der Literatur bekannt

- Quadratische Programmierung [60]
- Lineare Programmierung [37]
- Singulärwertzerlegung [61].

Die Berechnung der Singulärwerte garantiert keine Lösung. Ma und Kruth weisen darauf hin, dass die Singulärwertzerlegung für die Gewichte bei komplexen Modellen Instabilitäten erzeugt [61]. Lineare und Quadratische Programmierung hingegen sind zwei Lösungsmethoden, die auf die Berechnung der Gewichte von T-Spline-Flächen übertragbar sind.

Lineare Programmierung

Die Lineare Programmierung (LP) ist ein Optimierungsverfahren für die Berechnung eines allgemeinen Vektors $\mathbf{X}$ für eine lineare Zielfunktion unter einem Satz von Nebenbedingungen. Üblicherweise wird der Simplex-Algorithmus für die Berechnung einer Lösung verwendet [65] [14].
Nach der üblichen Definition der Linearen Programmierung[4] ist das Skalarprodukt

$$f(\mathbf{X}) := c_1 x_1 + \ldots + c_n x_n = \mathbf{C}^T \mathbf{X}$$

unter den Ungleichungs-Nebenbedingungen

$$\begin{aligned} a_{1,1}x_1 + \ldots + a_{1,n}x_n &\leq b_1 \\ &\vdots \\ a_{m,1}x_1 + \ldots + a_{m,n}x_n &\leq b_m \\ \Leftrightarrow A\,\mathbf{X} &\leq \mathbf{B} \end{aligned} \tag{4.21}$$

zu maximieren.

Ma und Kruth beschreiben eine mögliche Lösung von Gleichung (4.20) mittels Linearer Programmierung [60].

[4] Die Benennung der Matrizen und Vektoren entspricht dabei der Literaturdarstellung.

Das auf die Gewichtswerte von NURBS-Flächen angepasste Optimierungsproblem lautet

$$\left[\mathbf{0}^T\ \mathbf{1}^T\ \mathbf{1}^T\ \mathbf{1}^T\right] \cdot \begin{bmatrix} \mathbf{W} \\ \mathbf{P} \\ \mathbf{N} \\ \mathbf{S} \end{bmatrix} \tag{4.22}$$

unter einem linearen Gleichungssystem von Nebenbedingungen

$$\begin{bmatrix} M & -I & I & \\ I & & & -I \end{bmatrix} \cdot \begin{bmatrix} \mathbf{W} \\ \mathbf{P} \\ \mathbf{N} \\ \mathbf{S} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix} \tag{4.23}$$

für $p_i \geq 0$, $n_i \geq 0$ und $s_i \geq 0$. I ist die Einheitsmatrix. Die Voraussetzung $\mathbf{W} = \mathbf{1} + \mathbf{S}$ führt zu positiven Gewichten $w_i \geq 0$. Die Ungleichungen für die Nebenbedingungen werden nun mit $\mathbf{S}$ um eine sogenannte Schlupfvariable (*slack variable*) erweitert. Diese Schlupfvariable $\mathbf{S}$ sorgt dafür, dass die Gewichte w_i ausreichend entfernt von Null liegen und somit positiv ausfallen. Somit werden die Nebenbedingungen als System von Gleichungen (anstelle der Ungleichungen) beschrieben. Die Autoren führen in ihrer Arbeit verschiedene Fehlernormen für die Minimierung vor. Für die direkte Lösung verwenden sie die L1-Norm und nehmen die weiteren Schlupfvariablen $\mathbf{P}$ und $\mathbf{N}$ hinzu, um die Fallunterscheidung in der Betragsfunktion der L1-Norm (positiv oder negativ) zu formalisieren.

Quadratische Programmierung

Ein weiteres Optimierungsverfahren ist die Quadratische Programmierung (*QP*). Diese löst ein Minimierungsproblem mit einer quadratischen Zielfunktion. Hierfür werden, entsprechend der LP-Optimierung, Nebenbedingungen durch Gleichungs- und Ungleichungsbedingungen definiert [65].
Zu maximieren ist

$$f(\mathbf{X}) := \mathbf{X}^T Q \mathbf{X} + \mathbf{X}^T \mathbf{X} \tag{4.24}$$

mit den Nebenbedingungen

$$A\,\mathbf{X} \leq \mathbf{B}$$

für das System der Ungleichungen (vgl. Lineare Programmierung) und

$$E\,\mathbf{X} = \mathbf{D}$$

für ein Gleichungssystem mit den zusätzlichen Gleichheitsbedingungen.

Eine direkte Lösung für die Berechnung der isolierten Gewichte aus Gleichung (4.20) ist somit mit der QP-Methode möglich. Es gilt nun, die Gleichung

$$\mathbf{W}^T M^T M\,\mathbf{W} \tag{4.25}$$

unter den identischen Nebenbedingungen (4.23) der positiven Gewichte $w_i > 0$ zu minimieren.

Qualitativ ergeben sich ähnliche Ergebnisse wie mit der Optimierungsmethode der Linearen Programmierung.

4.2.4 Optimierung der Geometrie bei NURBS-Flächen

Die Gewichte $\mathbf{W}$ sind zu diesem Zeitpunkt bekannt. Aus dem Gleichungssystem (4.19) folgen nun die Tiefenwerte $\mathbf{C}$ der Kontrollpunkte direkt aus

$$B^T B\,\mathbf{C} = B^T Z B\,\mathbf{W}. \tag{4.26}$$

Dies sind ebenfalls lineare, überbestimmte Gleichungssysteme, daher eignet sich die Methode der kleinsten Quadrate für die Lösung.

4.3 Grundlagen der T-Spline-Flächenrückführung

Allgemein gut geeignete Ansatzgrade sind $K = 1$ (linear) oder $K = 3$ (kubisch) für die T-Spline-Funktion (2.7). Gerade Ansatzgrade (d. h. $K = 2, 4, \ldots$) werden in dieser Arbeit ausgeschlossen. Die Begründung hierzu folgt in Abschnitt 5.2. Höhere Grade sind zwar möglich, diese sind allerdings mit zusätzlichem Rechenaufwand behaftet. Die geometrische Optimierung beschränkt sich in dieser Arbeit lediglich auf die Verschiebung des Tiefenwerts eines Kontrollpunkts. Eine Erweiterung auf die x- und y-Komponenten führte bereits nach ersten Untersuchungen zu teilweise starken Verschiebungen der neu berechneten Kontrollpunkte innerhalb der T-Spline-Flächen und wird daher ausgeschlossen. Die Betrachtung der hier ignorierten x-y-Verschiebung von Kontrollpunkten kann somit in nachfolgenden Arbeiten untersucht werden.

Die Flächenrückführung mit T-Spline-Flächen entspricht weitgehend der Herangehensweise bei NURBS-Flächen. Das allgemeine Problem, angelehnt an die Minimierungsfunktion (4.2) und als Spezialisierung der NURBS-Modellmatrix (4.9) lautet als überbestimmtes Gleichungssystem

$$T\,\mathbf{C} = \mathbf{D}. \tag{4.27}$$

Durch linksseitige Ergänzung führt dies zu dem linearen Gleichungssystem

$$T^T T\,\mathbf{C} = T^T\,\mathbf{D} \tag{4.28}$$

mit

$$T := \begin{bmatrix} \frac{w_0}{f_0} \cdot N^K[\sigma_0,\tau_0](s_0,t_0) & \cdots & \frac{w_{n-1}}{f_0} \cdot N^K[\sigma_{n-1},\tau_{n-1}](s_0,t_0) \\ \vdots & \ddots & \vdots \\ \frac{w_0}{f_{m-1}} \cdot N^K[\sigma_0,\tau_0](s_{m-1},t_{m-1}) & \cdots & \frac{w_{n-1}}{f_{m-1}} \cdot N^K[\sigma_{n-1},\tau_{n-1}](s_{m-1},t_{m-1}) \end{bmatrix} \tag{4.29}$$

und

$$f_k := \sum_{l=0}^{n-1} w_l \cdot N^K[\sigma_l,\tau_l](s_k,t_k).$$

Die Flächenrückrührung mit T-Spline-Flächen unterscheidet sich wesentlich durch die getrennte Betrachtung von Standard-(ST) und Nicht-Standard-T-Spline-Flächen (NST) (Abschnitt 2.2.7). Standard-T-Spline-Flächen können mit linearen Modellfunktionen beschrieben und optimiert werden. Bei Nicht-Standard-T-Spline-Flächen zeigt sich die Nichtlinearität durch die Gewichtswerte $w_l \neq 1$. Diese werden, wie im vorherigen Abschnitt beschrieben, gesondert optimiert. Im Anschluss werden die Geo-

metrieoptimierung der Nicht-Standard-T-Spline-Flächen mittels linearer Optimierung und zusätzlich das Dragging-Verfahren vorgestellt.

4.4 Flächenrückführung mit Standard-T-Spline-Flächen

Aus der Matrix T in (4.29) folgt mit den Bedingungen für Standard-T-Spline-Flächen, d. h. $\forall i : w_i = 1$ und $\sum_i w_i N_i[\sigma_i, \tau_i](s,t) = 1$, die vereinfachte Form der Modellmatrix:

$$T_{ST} := \begin{bmatrix} N^K[\sigma_0,\tau_0](s_0,t_0) & \dots & N^K[\sigma_{n-1},\tau_{n-1}](s_0,t_0) \\ \vdots & \ddots & \vdots \\ N^K[\sigma_0,\tau_0](s_{m-1},t_{m-1}) & \dots & N^K[\sigma_{n-1},\tau_{n-1}](s_{m-1},t_{m-1}) \end{bmatrix} \tag{4.30}$$

Die anschließende Optimierung der Tiefenwerte an den Kontrollpunkten $\mathbf{C}$ erfolgt im Sinne der kleinsten Quadrate für das überbestimmte Gleichungssystem

$$T_{ST}\,\mathbf{C} = \mathbf{D}. \tag{4.31}$$

Dies ergibt das lineare Gleichungssystem

$$(T_{ST})^T\,T_{ST}\,\mathbf{C} = (T_{ST})^T\,\mathbf{D}. \tag{4.32}$$

Diese Optimierung entspricht der Flächenrückführung mit B-Spline-Flächen unter der Modellmatrix (4.7), allerdings mit lokalen Knotenwerten σ_i und τ_i. Eine Lösung des Gleichungssystems (4.32) wird durch beliebige Optimierungsverfahren für lineare Gleichungssysteme erreicht.

4.5 Flächenrückführung mit Nicht-Standard-T-Spline-Flächen

Angelehnt an Heidrich et al. [37] lässt sich das Gleichungssystem (4.19) für die (allgemeine) Modellmatrix von Nicht-Standard-T-Spline-Flächen (4.29) analog schreiben als

$$\begin{bmatrix} T^TT & -T^TZT \\ 0 & M \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \mathbf{W} \end{bmatrix} = \mathbf{0} \tag{4.33}$$

mit

$$M := T^TZ^2T - (T^TZT)(T^TT)^{-1}(T^TZT).$$

In den weiteren Schritten werden die Gewichte und die Geometrie für die Optimierung der Fläche analysiert.

4.5.1 Optimierung der Gewichte bei Nicht-Standard-T-Spline-Flächen

Aus (4.33) ergeben sich die Gewichte als

$$M\,\mathbf{W} = \mathbf{0}. \tag{4.34}$$

Die Berechnung der Gewichte ist bereits in Abschnitt 4.2.3 beschrieben.

Hieraus ergeben sich zwei Fälle für das Ergebnis von Gleichung (4.34):

- Es findet sich eine nichttriviale Lösung für die Gewichtswerte, d. h. $\exists i : w_i \neq 0$. Dann ist die gefundene Lösung die optimale Lösung. Eine gute Lösung besteht dabei aus (positiven!) Gewichten, die nahe bei 1 liegen
- Es findet sich lediglich die triviale Lösung, d. h. $\forall i : w_i = 0$. Die Berechnung einer weiteren Lösung ist nicht möglich. Die Gewichte werden als $\mathbf{W} := \mathbf{1}$ festgesetzt.

Eine getrennte Anpassung der Gewichte in den eingesetzten Beispielen aus den Kapiteln 3 und 7 hat zu keinen nennenswerten Vorteilen geführt. Die erzielte Werte liegen durchgehend in einem sehr engen Intervall um 1, so dass keine Neuberechnung der Gewichte erforderlich ist. In weiteren Beispielen sind allerdings davon abweichende Ergebnisse möglich.

4.5.2 Optimierung der Geometrie bei Nicht-Standard-T-Spline-Flächen

Da die Gewichte $\mathbf{W}$ als Ergebnis aus Gleichung 4.34 bekannt sind, folgen aus dem Gleichungssystem (4.33) die geometrischen x-,y- und z-Komponenten der Kontrollpunkte. Das folgende Gleichungssystem beschreibt somit die Flächenrückführung bei Nicht-Standard-T-Spline-Flächen für Bathymetrien:

$$T^T T \, \mathbf{C} = T^T Z T \, \mathbf{W}. \tag{4.35}$$

Die aus den Multiplikationen entstehenden Produktmatrizen haben dabei eine Reihendimension gleich der Anzahl der Messpunkte und eine Spaltendimension gleich der Anzahl der Kontrollpunkte im T-Netz (vgl. Abschnitt 3.2.4).

4.6 Dragging-Verfahren für T-Spline-Flächen

Berkhahn und Kaapke greifen auf ein Approximationsschema für die Anpassung einer B-Spline-Fläche an eine Menge von Messpunkten zurück [10] [43]. Aus diesen Arbeiten stammt die Benennung als „Dragging-Verfahren“ (und synonym hierzu „Dragging-Algorithmus“ oder „Dragging-Optimierung“). Der Algorithmus arbeitet progressiv, d. h. er wird mit einer Anzahl an Schritten bzw. bis zum Erreichen eines optimierenden Abbruchkriteriums (z. B. Unterschreiten eines maximal erlaubten Fehlers) durchgeführt. In jedem Schritt werden die Tiefenwerte der Kontrollpunkte verschoben und insgesamt verringern sich die Differenzfehler zwischen der originalen Geometrie und der erzeugten Fläche. Voraussetzung hierfür ist allerdings, dass die Referenzfläche kontinuierlich ist, d. h. zu jedem Parameterpunkt existiert ein Flächenpunkt (andernfalls ist der lokale Differenzvektor nicht bestimmt).

Lin und Zhang [57] bzw. Zhao et al. [97] beschreiben einen Algorithmus mit vergleichbarem Ablauf zur Approximation eines T-Netzes im Zusammenhang mit der Verarbeitung von Digitalbildern.

4.6.1 Ablauf

Im eigentlichen Algorithmus sind:

- k die Progressionsstufe, d. h. $k = 0$ für das initiale T-Netz
- $\mathbf{c_i^k} \in \mathbb{E}^3$ als i-ter Kontrollpunkt ($i = 0, \ldots, n-1$) im T-Netz der Stufe k. Genauer: die geometrische Komponente des Kontrollpunkts.

- $(s_i, t_i) \in \mathbb{R}^2$ das auf jeder Stufe k identische Parameterpaar für den Kontrollpunkt $\mathbf{c_i^k}$

- $\mathbf{t^k}(s_i, t_i) \in \mathbb{E}^3$ als Punkt einer T-Spline-Fläche der k-ten Progressionsstufe

- $\mathbf{s}(s_i, t_i) \in \mathbb{E}^3$ als Punkt der Referenzfläche am Parameterpaar (s_i, t_i)

- $\boldsymbol{\delta_i^k} := \mathbf{s}(s_i, t_i) - \mathbf{t^k}(s_i, t_i)$ als Differenzvektor (*Dragging-Vektor*) der Stufe k zwischen einem Punkt auf der Referenzfläche und einem Punkt der T-Spline-Fläche bei identischem Parameterpaar

Die Beziehungen der genannten Punkte und Vektoren sind in Abbildung 4.2 ersichtlich. Der Algorithmus für das vollständige Dragging-Verfahren weist keine Überraschungen auf und berechnet progressiv T-Netze (Algorithmus 4).

Eingabe: T-Netz auf Stufe 0 mit den Kontrollpunkten $C^k := \{\mathbf{c_i^k}\}$
begin
  repeat
    foreach *Kontrollpunkt* $\mathbf{c_i^k}$ **do**
      – Berechne $\boldsymbol{\delta_i^k} := \mathbf{s}(s_i, t_i) - \mathbf{t^k}(s_i, t_i)$
      – Berechne $\mathbf{c_i^{k+1}} := \mathbf{c_i^k} + \boldsymbol{\delta_i^k}$
    end
  until *Abbruchkriterium erfüllt*;
  – Erzeuge das T-Netz der Stufe $k+1$ als $C^{k+1} := \{\mathbf{c_i^{k+1}}\} = C^k + \Delta^k$
end
Ausgabe: Optimiertes T-Netz für eine T-Spline-Fläche

Algorithmus 4 : Dragging-Verfahren

Die Menge der Kontrollpunkte nach jedem Schritt k ist $C^k := \{\mathbf{c_i^k}\}$, und die berechneten zugehörigen Dragging-Vektoren sind $\Delta^k := \{\boldsymbol{\delta_i^k}\}$. Abbildung 4.3 zeigt die Verschiebung nach einem Einzelschritt. Auch hier wird das Verfahren auf die Verschiebung der Tiefenwerte an den Kontrollpunkten reduziert. Das Abbruchkriterium kann eine feste Anzahl an Schritten oder flexibel durch ein Fehlermaß gegeben sein. Ein gutes Merkmal stellt der längste Dragging-Vektor jeder Stufe dar: dieser drückt den maximalen Fehler in der aktuellen Stufe der Approximation aus und somit wird bei Unterschreiten eines Wertes der Algorithmus beendet.

Lin und Zhang [57] berechnen die Verschiebung eines Kontrollpunkts $\mathbf{c_i^k}$ explizit anhand der Menge der Kontrollpunkte, deren T-Spline-Basisfunktionen > 0 ergeben. Die vorgestellte OpenCL-Implementierung verzichtet aus Effizienzgründen auf diesen zusätzlichen Test, da Kontrollpunkte mit dem Vorfaktor 0 in der Flächenberechnung ohnehin keinen Einfluss auf den Flächenverlauf nehmen.

Die Autoren weisen die allgemeine Konvergenz des Verfahrens nach. Aus praktischer Sicht wird eine Limitierung der Verschiebung empfohlen, z. B. die Halbierung der maximalen Länge der Dragging-Vektoren der vorherigen Stufe. Dieser Ansatz konvergiert nachweislich.

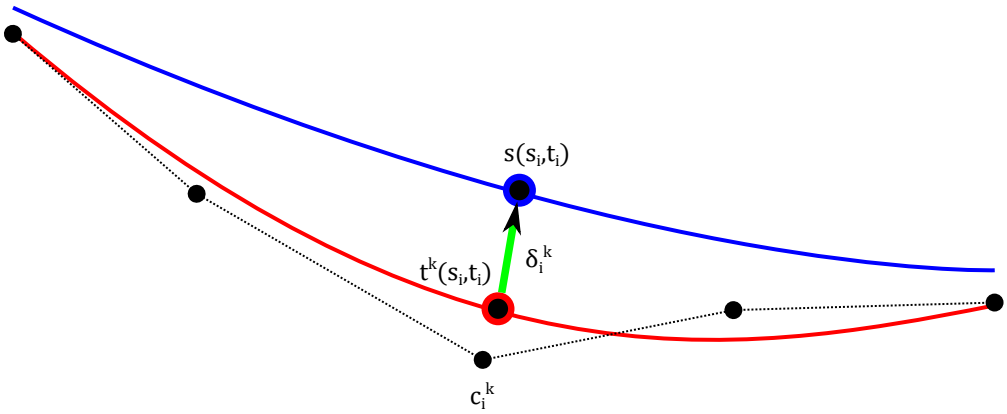


Abbildung 4.2: Dragging-Verfahren zwischen der Referenzfläche (blau) und der generierten T-Spline-Fläche (rot). $\mathbf{c_i^k}$ ist der Kontrollpunkt mit dem Parameterpaar (s_i, t_i), $\mathbf{t^k}(s_i, t_i)$ ein Punkt auf der T-Spline-Fläche und $\mathbf{s}(s_i, t_i)$ ein Punkt auf der Referenzfläche. $\boldsymbol{\delta_i^k}$ gibt die Verschiebung des Kontrollpunkts für die nachfolgende Iteration an.

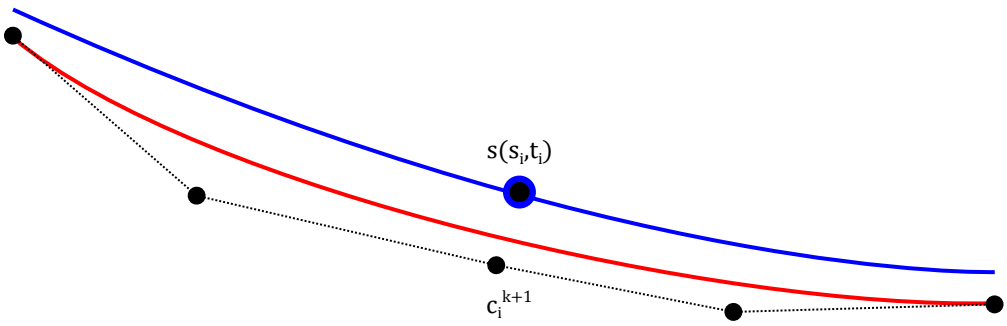


Abbildung 4.3: Fortschritt im Dragging-Verfahren: Der Kontrollpunkt $\mathbf{c_i^k}$ wird um den Vektor $\boldsymbol{\delta_i^k}$ verschoben und die T-Spline-Fläche nähert sich der Referenzfläche an.

4.6.2 Bewertung des Dragging-Verfahrens

Das Dragging-Verfahren ist eingebettet in das Approximationsschema zur progressiven Verfeinerung von Kontrollpunktnetzen nach [66] (Abschnitt 4.1). Vorteile bietet das Dragging-Verfahren hinsichtlich

- der robusten Konvergenz und somit einer wählbaren Approximationsgüte, und
- der Skalierbarkeit bezogen auf die Netzgröße.

Im Vergleich mit geschlossenen[5] Optimierungsverfahren stellt sich heraus, dass die tatsächliche Berechnungsdauer nicht deterministisch ist, d. h. der Zeitpunkt der Konvergenz ist unbekannt. Zu beachten ist außerdem, dass das Dragging-Verfahren bei ungünstigen Konstellationen im Verlauf der Optimierung ein starkes Schwingungsverhalten aufweisen kann. In der Praxis zeigt sich oftmals ein Unter- und Überschwingen des Flächenverlaufs bezogen auf die bathymetrische Referenzfläche. Dies führt zu einem Vorzeichenwechsel der Dragging-Vektoren. Die Längenhalbierung der Dragging-Vektoren schränkt dieses Verhalten ein.

4.7 Zwischenstand

Kern der linearen Optimierung sind Matrixoperationen. Ausgehend von der Modellmatrix T ist eine effiziente Berechnung des überbestimmten, linearen Gleichungssystems (4.35) notwendig. Die Vorberechnung der Modellmatrix ist hochgradig parallelisierbar (Abschnitt 6.3.4). Allerdings ergeben sich bei sehr großen Matrizen technische und folglich numerische Probleme hinsichtlich der Speichergröße, der Rechenzeit und der Genauigkeit der Lösung.

Das umgesetzte Approximationsschema führt in jeder Iteration eine Reihe von Arbeitsschritten durch. Innerhalb dieser besteht ein hohes Maß an wiederkehrenden Operationen. Jeder der folgenden Schritte besitzt für sich betrachtet bereits einen hohen Optimierungsbedarf:

- Herleitung der lokalen Knotenwerte
- Erzeugen einer T-Spline-Fläche
- Lokale Zellverfeinerung
- Lineare Geometrieoptimierung mit der Methode der kleinsten Quadrate
- Geometrieoptimierung mit Dragging
- Berechnen von kumulierten Differenzfehlern.

Es zeigt sich, dass mit zunehmender Netzgröße der Rechenbedarf überproportional steigt. Die meisten der o. g. Schritte greifen jeweils auf sehr kleine Problemräume zurück. Beispielsweise werden die Knotenwertherleitung oder das Dragging-Verfahren isoliert auf einzelnen Kontrollpunkten angewendet. Das folgende Kapitel 5 präsentiert eine optimierte Datenstruktur für T-Netze. Diese stellt die Grundlagen der angepassten Algorithmen für die GPGPU-unterstützte Implementierung in Kapitel 6 dar.

Ein direkter Vergleich zwischen der linearen Optimierung und dem Dragging-Algorithmus anhand einer realen Bathymetrie findet schließlich in Kapitel 7 statt.

[5] Ein mathematisches Verfahren mit eindeutiger Lösung und deterministischer Rechenzeit.

5 T-Netze und Operationen

T-Netze sind grafisch darstellbare Kontrollpunktnetze für T-Spline-Flächen. Die hier vorgestellten Erweiterungen der wohl erforschten R-Bäume zeichnen sich dadurch aus, modifizierende Operationen auf T-Netzen effizient durchzuführen. Der Schwerpunkt liegt dabei auf der Beschreibung von grundlegenden Konzepten (z. B. räumliche Suche), die eine auf diesen Kenntnissen basierende GPGPU-Entwicklung ermöglichen.

5.1 Anforderungen

Ein T-Netz ist ein Zellkomplex, der grafisch darstellbar ist. Es soll die im Folgenden vorgestellten Aspekte von T-Spline-Flächen abdecken.

5.1.1 Regelmäßige T-Netze

Ein regelmäßiges, d-dimensionales T-Netz ist ein Kontrollpunktnetz aus Polyedern (bezeichnet als Zellen) für die Kontrollpunkte im Raum $\mathbb{R}^d$. Die Dimension des Parameterraums ist $\leq d$, üblicherweise jedoch $d-1$ (z. B. zweidimensionale Flächen im dreidimensionalen Raum). Bei T-Spline-Flächen sind die Polyeder somit allgemeine Parallelogramme, in der Praxis allerdings orthogonale, achsenparallele Rechtecke (Abbildung 5.1). An den Kreuzungspunkten der Polyederkanten liegen die Kontrollpunkte. Im Unterschied zu den vollbesetzten Spline-Kontrollpunktgittern ist bei T-Netzen keine Reihen-Spalten-Adressierung definiert. Ein eindimensionaler Index i listet die Kontrollpunkte P_i (Definition s. u.) auf.

Sederberg et al. definieren T-Netze als Menge an Kanten zwischen Kontrollpunkten, weisen den Kanten jedoch zusätzliche Knotenintervalle zu [77]. Die Knotenintervalle geben die Differenzen der lokalen Knotenwerte vor. Die Intervalle an den Kanten sind jedoch nicht zwingend erforderlich, sofern die Parameterpaare der Kontrollpunkte bekannt sind. Der Dualismus zwischen Knotenintervallen und Knotenwerten wird in Abschnitt 5.2.1 beschrieben.

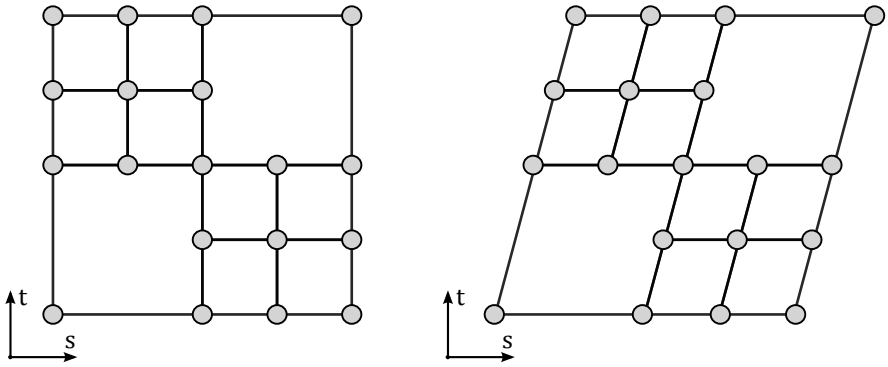


Abbildung 5.1: T-Netze im Parameterraum mit Rechtecken (links) und Parallelogrammen (rechts)

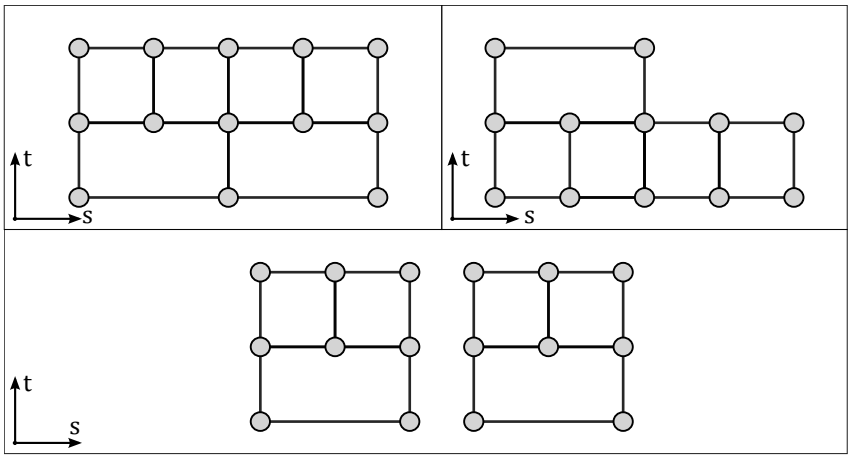


Abbildung 5.2: Beispiele für Topologien von T-Netzen: rechteckig (oben links), L-förmig (oben rechts), zusammenhanglos (unten)

5.1.2 Generische Datenstruktur

Die zu entwickelnde Datenstruktur enthält die Kontrollpunkte P_i. Diese bestehen aus den Parameterpaaren der Kontrollpunkte im T-Netz $\mathbf{p_i} := (s_i, t_i) \in \mathbb{R}^2$, den geometrischen Koordinaten der Kontrollpunkte $\mathbf{c_i} := (x_i, y_i, z_i) \in \mathbb{E}^3$ (allgemein $\mathbb{E}^d$) und ihren Gewichten w_i. Ein Kontrollpunkt ist somit ein mehrdimensionaler, generischer Vektor. T-Netze stellen eine abstrakte Datenstruktur dar, da sie Parameter- und Geometrieraum verbinden und Änderungen gegenseitigen Einfluss haben. Optional können lokale Knotenwerte gespeichert werden. Da diese allerdings nach modifizierenden Netzoperationen ungültig werden, müssen sie aktualisiert werden. Änderungen am T-Netz ähneln daher prinzipiell denen an NURBS-Flächen. Bei diesen haben Änderungen in der Geometrie ebenfalls einen gegenseitigen Einfluss auf die Netztopologie und die Knotenwerte (Abschnitt 2.2.8).

5.1.3 Topologie

Prinzipiell können T-Netze deutlich von einer rechteckigen Form für das gesamte Kontrollpunktgitter abweichen. So sind beispielsweise globale L- oder T-förmige Netztopologien möglich (Abbildung 5.2). Genauso sind zusammenhanglose Flächen vorstellbar. Generell lässt sich sagen, dass die allgemeine und freie Verbindung zweier T-Netze mit rechteckiger Grundfläche zu einer beliebigen Topologie führen kann (Abschnitt 2.2.8). Daraus ergeben sich theoretisch keine weiteren Einschränkungen, da lokale Knotenwerte ohnehin nur den einzelnen Kontrollpunkten zugewiesen sind.

Der Wertebereich einer T-Spline-Fläche ergibt sich aus dem T-Netz. Abbildung 5.3 zeigt ein Beispiel für einen Flächenpunkt, der innerhalb des parametrischen Einflussgebiets eines Kontrollpunkts liegt (d. h. es existieren Basisfunktionen ≥ 0), allerdings außerhalb des Definitionsbereichs der T-Spline-Fläche. Da die Netze beliebige Formen annehmen, und der Wertebereich stets innerhalb der Netzfläche liegt, muss ein effektiver Algorithmus den häufigen Lagetest eines Punkts, d. h. liegt ein Punkt innerhalb oder außerhalb des Wertebereichs, umsetzen. In der oben aufgeführten Definition bestehen T-Netze aus einer Menge an achsenparallelen Rechtecken, so dass sich räumliche Datenstrukturen aus der algorithmischen Geometrie eignen (Abschnitt 5.5).

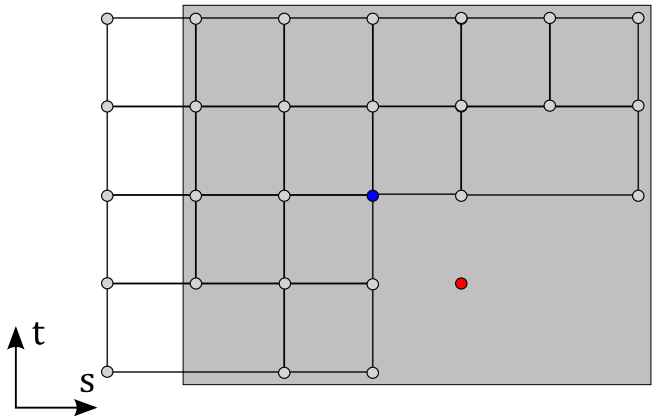


Abbildung 5.3: Der abgebildete rote Punkt liegt innerhalb des parametrischen Einflussgebiets (grau) des Kontrollpunkts (blau). Dies ist das durch die Knotenwerte induzierte Gebiet des Kontrollpunkts.

5.1.4 Erweiterte T-Netze

Werden Kanten mit Nullintervallen berücksichtigt, d. h. es existieren Kontrollpunkte mit identischem Parameterwert, so kann man den parametrischen Raum des Kontrollpunktnetzes in einen höheren Bildraum $I \subseteq \mathbb{N}^d$ einbetten (Abbildung 5.4). Auf diesem Weg wird die funktionale Abbildung von Bild- auf Parameterwerten und somit auf Geometriekoordinaten sichergestellt. In den Originalarbeiten finden Nullintervalle Verwendung, allerdings werden diese im Kontext der hier beschriebenen Geländemodellierung nicht praktisch verwendet. Eine genaue Benennung derartiger Kontrollpunktnetze ist nicht bekannt, sie können als erweiterte oder eingebettete T-Netze bezeichnet werden.

5.2 Lokale Knotenwerte

Die Herleitung der lokalen Knotenwerte ist eine der wesentlichen Operationen auf T-Netzen. Sederberg et al. beschreiben die Herleitung der dualen Knotenintervalle [77]. Abweichend davon werden die Knotenwerte in dieser Arbeit direkt aus dem parametrischen Koordinatenraum des T-Netzes bestimmt. Die Beschreibung erfolgt für ungerade Grade K der T-Spline-Basisfunktion (linear oder kubisch). Gerade Ansatzgrade ($K = 2, 4, \ldots$) können ebenfalls modelliert werden. Hierbei erfolgt die Herleitung der Knotenwerte allerdings nicht anhand der Kanten im T-Netz. Stattdessen ergeben sich diese aus den parametrischen Mittelpunkten der Zellen im T-Netz. Hierbei entfällt allerdings die intuitive Modellierung durch die Zuordnung der Knotenwerte über die Parameterwerte der Kontrollpunkte. Zusätzlich werden die drei Bedingungen für valide T-Netze eingeführt.

5.2.1 Dualität von Knotenintervallen und Knotenwerten

Sederberg et al. [77] verwenden ursprünglich die Notation von (relativen) Knotenintervallen, d. h. jeder Kante im T-Netz wird ein Intervallwert ≥ 0 zugewiesen (Abbildung 5.5). Die lokalen Knotenwerte ergeben sich somit aus der Summierung der Intervallwerte. Der Sonderfall eines Nullintervalls führt ggf. zu einer Unstetigkeit der Fläche, da hier Parameterwerte folglich mehrfach auftreten. Die prinzipielle Her-

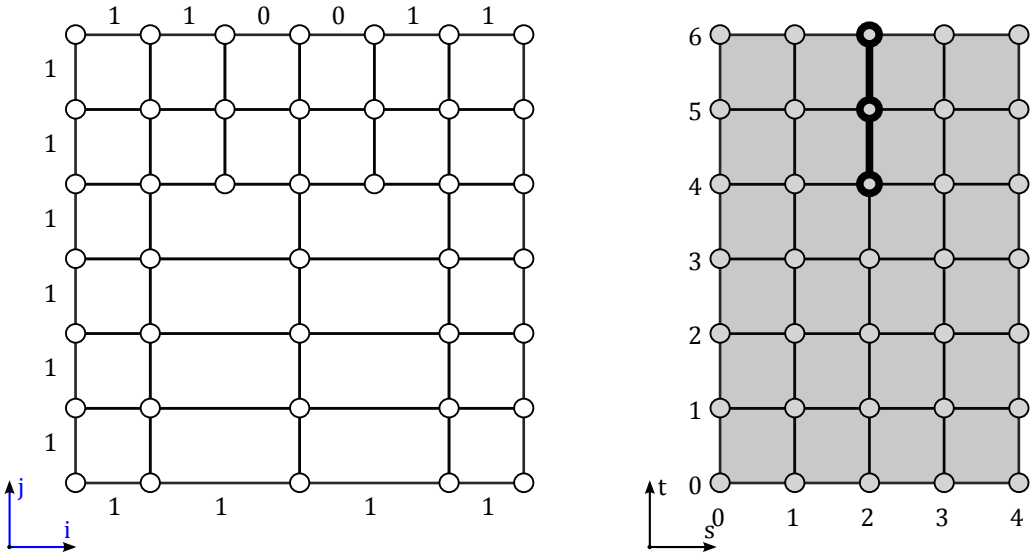


Abbildung 5.4: Darstellung von Bildraum (links) und Parameterraum (rechts) bei einem T-Netz mit Intervallwerten zu der T-Spline-Fläche in Abbildung 2.6. Die horizontalen Null-Kanten kollabieren im Parameterraum.

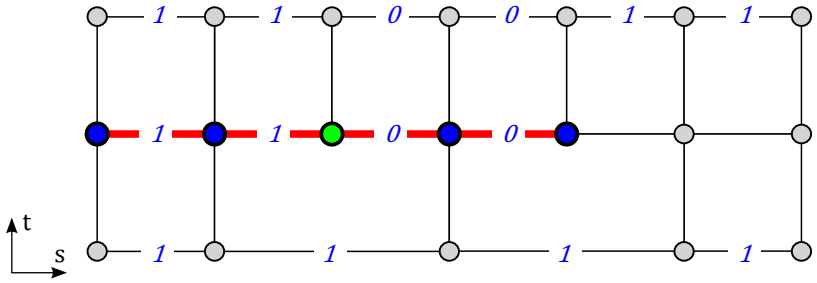


Abbildung 5.5: Darstellung der Knotenintervalle in einem T-Netz für einen Kontrollpunkt (grün). Die horizontalen Knotenintervalle lauten [1,1,0,0], die Knotenwerte somit [0,1,2,2,2].

leitung der Knotenintervalle erfolgt durch die gleichen Schnittberechnungen in jede Richtung des T-Netzes wie bei der Berechnung der Knotenwerte (s. u.). Absolute Parameterwerte für Kontrollpunkte sind allerdings gegenüber der Intervallnotation im Vorteil, da hierbei eine intuitive Modellierung im Parameterraum ermöglicht wird. Die relative Intervallnotation erfordert jedoch ein zusätzliches Umrechnen auf absolute Parameterwerte.

5.2.2 Knotenwerte bei ungeradem Grad

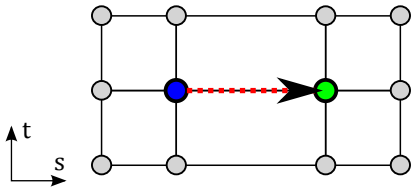
Für eine T-Spline-Fläche vom Grad K besteht der lokale Knotenwert-Vektor aus $K+2$ Komponenten in jede Richtung[1]. Die Bestimmung der Knotenwerte bei ungeradem Grad am Parameterpunkt p_i erfolgt über Schnittberechnungen entlang der Parameterachsen. Bei einem zweidimensionalen T-Netz werden jeweils Strahlen in West- $(-x)$ bzw. Ostrichtung $(+x)$ und Süd- $(-y)$ bzw. Nordrichtung $(+y)$ ausgesendet. Daraus resultieren die nächsten $\frac{K+1}{2}$ Schnitte, d. h. mit linearem Grad jeweils ein Schnitt, mit kubischem Grad zwei Schnitte, usw.

[1] Vereinfacht ausgedrückt als Nord, Süd, Ost, West.

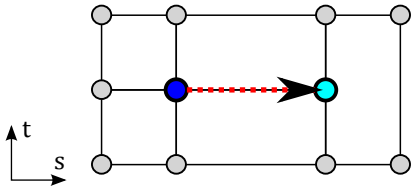
Jeder Schnitt erzeugt Schnittpunkte $\mathbf{q_k} := (s_k, t_k)$ mit $k = -\frac{K+1}{2}, \ldots, -1, 1, \ldots, \frac{K+1}{2}$. Die lokalen horizontalen Knotenwerte σ_i werden aus den entsprechenden Komponenten der Schnittpunkte konstruiert, Somit gilt $\sigma_i := [s_{-\frac{K+1}{2}}, \ldots, s_{-1}, s_i, s_1, s_{\frac{K+1}{2}}]$. Die vertikalen Knotenwerte τ_i folgen entsprechend.

Für einen Schnittpunkt $\mathbf{q_k}$ des Strahls mit einer Zelle können drei Fälle eintreten (dargestellt für rechteckige T-Netze und horizontalen Schnitt).

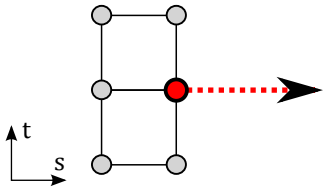
1. Der Strahl trifft direkt auf einen Kontrollpunkt.



2. Der Strahl trifft eine orthogonale (d. h. vertikale) Kante. In dem Fall ist $\mathbf{q_k}$ der Schnittpunkt zwischen Strahl und Kante.



3. Es gibt keinen Schnitt. Letztlich trifft dies für Kontrollpunkte am Rand zu.



Insgesamt sind für jede Richtung $\frac{K+1}{2}$ Schnitte nötig. Falls weniger Schnitte sind, so gibt es weitere Strategien für die Behandlung der Unstetigkeit am Rand. Hier werden fehlende Knotenwerte durch mehrfaches Auftreten korrigiert. Eine Alternative hierzu beschreibt [42].

Ein Beispiel für eine kubische T-Spline-Fläche mit einer Randunstetigkeit zeigt Abbildung 5.6. Für die lokalen Knotenwerte gilt $\sigma_i := [0, 0, 1, 2, 3]$ und $\tau_i := [2, 3, 4, 5, 6]$. Die resultierende Fläche weist oberhalb der Mitte einen Knick auf (erkennbar an der Reflektion der simulierten Beleuchtung).

Insgesamt ergibt sich die Notwendigkeit, Schnittberechnungen in T-Netzen zwischen einem Strahl und Rechtecken effizient durchzuführen. Eine hochgradig performante Datenstruktur für die Identifikation von geschnittenen Zellen sind R-Bäume (Abschnitt 5.5.1). Durch geschickte Suchanfragen werden die geschnittenen Objekte (d. h. Punkte und Kanten) direkt und in logarithmischer Suchzeit (bezogen auf die Anzahl der Zellen) bestimmt.

Ein T-Netz ist statisch, falls sich seine Topologie und die Positionen der Kontrollpunkte nicht ändern. Die geometrischen Komponenten der Kontrollpunkte dürfen verändert werden. In einem statischen T-Netz können die lokalen Knotenwerte vorab

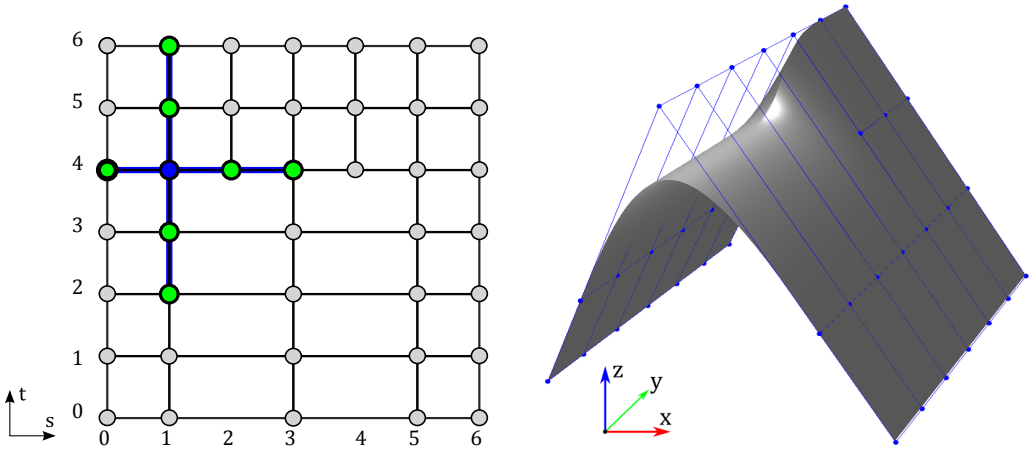


Abbildung 5.6: Herleitung von kubischen Knotenwerten ausgehend von einem Kontrollpunkt (blau) über die benachbarten Kontrollpunkte (grün) und Darstellung der interpolierten T-Spline-Fläche

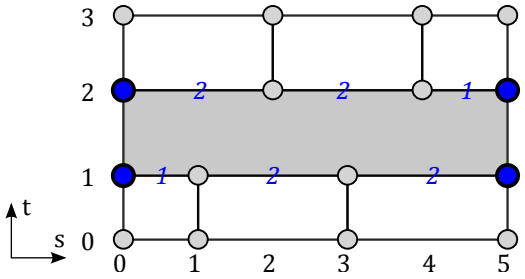


Abbildung 5.7: Darstellung der ersten Regel: Die Summe gegenüberliegender (horizontaler) Knotenintervalle ist immer identisch, somit bilden die Kontrollpunkte ein Rechteck im Parameterraum.

berechnet und den Kontrollpunkten fest zugewiesen werden. Dynamische T-Netze erfordern allerdings eine Neuberechnung der Knotenwerte nach jeder Änderung.

5.2.3 Regeln für T-Netze

Sederberg nennt drei Bedingungen, die innerhalb eines validen T-Netzes stets gelten müssen [77]:

1. Die Summe der Knotenintervalle auf gegenüberliegenden Seiten jeder T-Netz-Zelle muss gleich sein. Übertragen auf die Knotenwerte bedeutet dies, dass Eckpunkte der Zellen stets ein Rechteck mit gegenüberliegenden Kanten bilden. Die innere Verteilung der Kontrollpunkte ist beliebig (Abbildung 5.7).

2. Falls ein Punkt auf einer Kante mit einem gegenüberliegenden Punkt verbunden werden kann, ohne die erste Regel zu verletzen, so müssen diese verbunden werden. Die bestehende Zelle wird in zwei neue unterteilt. Konkret besitzen die beiden Punkte eine gleiche Parameterkomponente, so dass bspw. bei identischen horizontalen Komponenten eine vertikaler Teilung erfolgt (Abbildung 5.8).

3. Eine horizontale Kante einer T-Zelle wird genau dann mit einem neuen Kontrollpunkt Q aufgeteilt (d. h. lokal verfeinert), falls Q auf einer Linie mit den

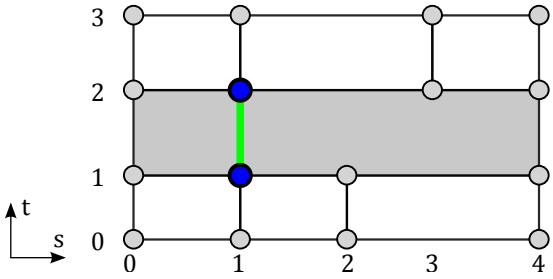


Abbildung 5.8: Darstellung der zweiten T-Netz-Regel: Gegenüberliegende Kontrollpunkte (blau) mit identischem Parameterwert müssen durch eine Kante (grün) verbunden werden.

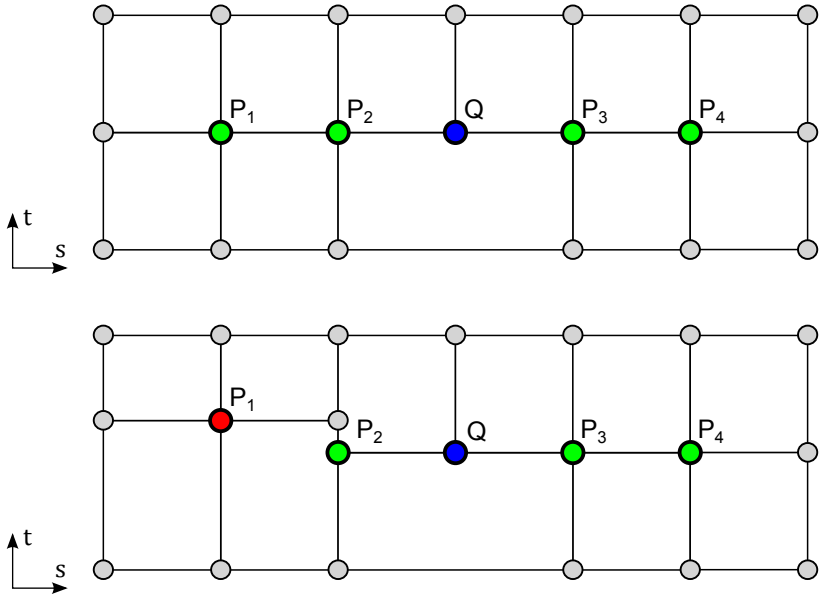


Abbildung 5.9: Darstellung der dritten Regel für die lokale Verfeinerung: Der Kontrollpunkt Q darf nur bei identischen vertikalen t-Komponenten von P_1 bis P_4 eingefügt werden (oben). P_1 verletzt Regel 3 (unten).

benachbarten Kontrollpunkten liegt und die vertikalen Komponenten identisch sind. Die Teilung von vertikalen Kanten erfolgt analog (Abbildung 5.9).

Sederberg et al. setzen diese drei Regeln für die Verfeinerung von T-Netzen voraus [77]. Die Konstruktion eines validen T-Netzes nach den o. g. Regeln führt somit erst zu einer korrekten Verteilung der lokalen Knotenwerte.

Regel 1 und 2 sind für die Konstruktion eines T-Netzes relevant und müssen in jeder Phase beachtet werden. Die dritte Regel stellt den Grundsatz der Netzverfeinerung dar, auf sie wird in Abschnitt 5.4.4 Bezug genommen.

5.3 Hängende Knoten

Ein qualitatives Problem von T-Spline-Flächen ergibt sich aus den in der FEM[2]-Netzzerlegung bekannten „hängenden Knoten“. In einem T-Netz kann eine große Zelle deutlich kleinere Nachbarn haben. Dies führt zu nicht-geradlinigen Verzerrungen in

[2]Finite-Elemente-Methode

den Verläufen der Isoparameterkurven einer Fläche. Dieser Effekt hat einen spürbaren Einfluss auf die Qualität der Flächenrückführung im Gebiet der hängenden Knoten, da die lokale Dichte der Kontrollpunkte sprunghaft zunimmt.

Für die angestrebte lokale Verfeinerung empfiehlt es sich, zwischen den Zellen maximal eine 1 : 2 -Nachbarschaft zu konstruieren. Eine 1 : n -Nachbarschaft bezeichnet hierbei einen Wechsel der Kantenlängen von benachbarten Zellen von einem konstanten δ_s oder δ_t zu $n \cdot \delta_s$ (bzw. $n \cdot \delta_t$) im Parameterraum. Dies erfordert jeweils Verfeinerungen der größeren Zellen.

Abbildung 5.10 zeigt ein Beispiel einer vollkommen ebenen Fläche mit schrittweise erhöhten Nachbarschaftsbeziehungen (1 : 1, 1 : 2 und 1 : 4). In der x-y-Aufsicht ist eine Schar in horizontaler Parameterrichtung verlaufender Isoparameterkurven dargestellt. Die Verzerrungen entlang der vertikalen Flächenrichtung nehmen mit zunehmender Nachbarschaft zu. Aus diesem Grund werden die T-Netze in dieser Arbeit stets mit einer 1 : 2 -Nachbarschaft erzeugt. Findet sich in einem T-Netz eine Zelle mit höherer Nachbarschaftsdichte, so wird das T-Netz innerhalb dieser Zelle erneut unterteilt.

5.4 Erforderliche Operationen auf T-Netzen

Ausgangspunkt für die Wahl einer effektiven Datenstruktur für T-Netze ist die Anwendung der in diesem Kapitel aufgelisteten Operationen und Anforderungen. Vereinzelt beruhen diese Operationen aufeinander, wie beispielsweise die Identifikation von Zellnachbarschaften eine Unterroutine für die Schnittberechnung darstellt, die wiederum wesentlich für die Herleitung der lokalen Knotenwerte ist. Der Fokus der Bewertung liegt hierbei auf den u. g. Operationen, die mit dynamischen Netzen in der Reduktion von bathymetrischen Flächen häufig auftreten. Bei statischen Netzen stehen andere Aspekte im Vordergrund (bspw. räumliche Suchanfragen).
Nach ihrer Anwendungshäufigkeit geordnet sind dies:

- die kombinierte Speicherung der Topologie im Parameterraum und der räumlichen Geometrie der T-Spline-Fläche
- die Suche von Kontrollpunkten und T-Netz-Zellen und die Bestimmung direkter Nachbarschaftsbeziehungen
- die Herleitung der Knotenwerte durch Schnittberechnungen im Parameterraum
- die lokale Verfeinerung von bestehenden Netzen
- die Verbindung von Teilflächen, insbesondere die allgemeine Verbindung von beliebigen T-Netzen
- die lokale (oder ggf. globale) Vergröberung von T-Netzen
- die Modellierung mit Null-Knotenintervallen und die Einbettung eines T-Netzes in einen übergeordneten Bildraum I
- die Extraktion von Bézier-Flächen
- die Erweiterbarkeit von zweidimensionalen auf d-dimensionale T-Polyeder für Volumengeometrien oder Raum-Zeit-Modelle

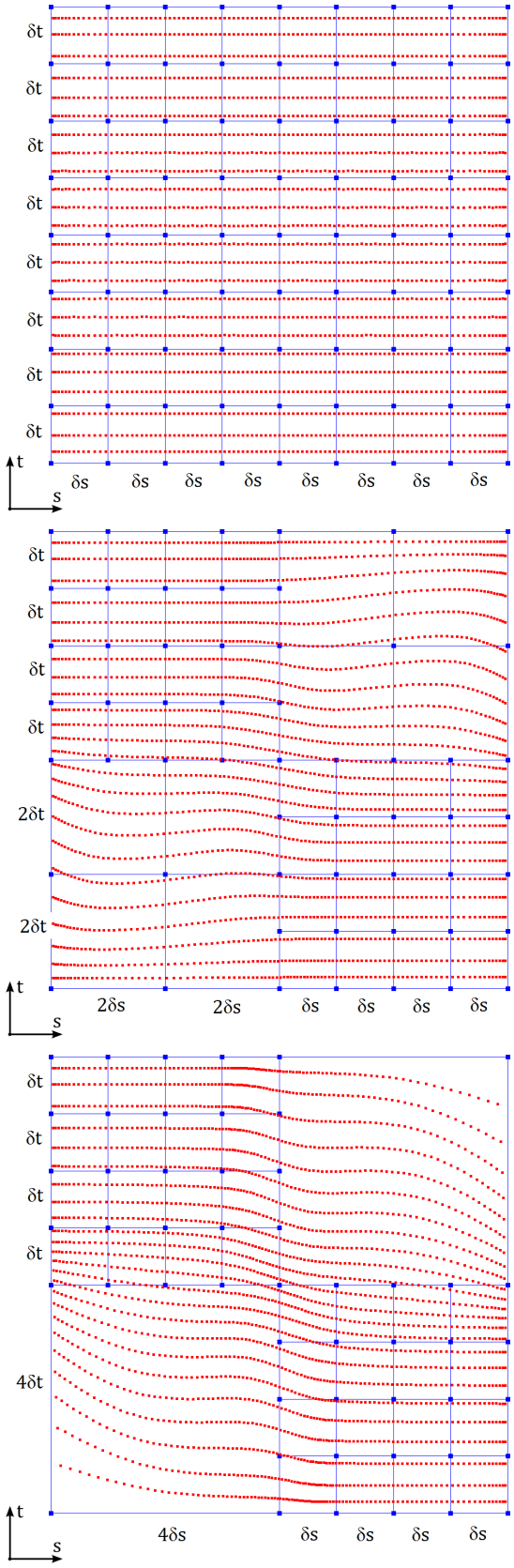


Abbildung 5.10: Verzerrung der Isoparameterkurven bei 1 : 1, (oben), 1 : 2 (Mitte) und 1 : 4-Nachbarschaften (unten)

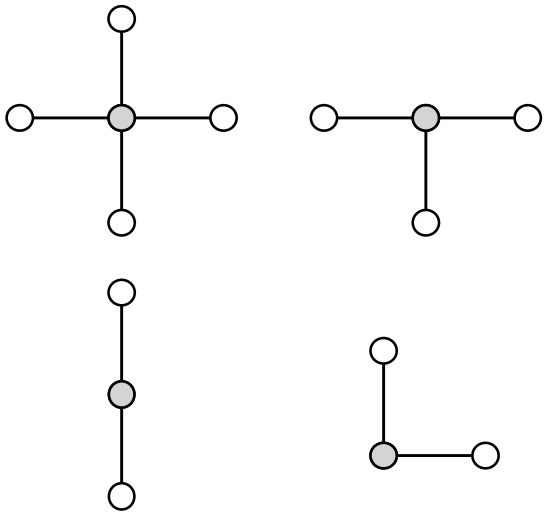


Abbildung 5.11: Mögliche Knotenformen in T-Netzen (Kreuzung, T-Punkt, I-Punkt und L-Punkt)

5.4.1 Parameterraum und Topologie

Ein T-Netz verkörpert den topologischen Zusammenhang einer T-Spline-Fläche. Üblicherweise haben die Kontrollpunktnetze eine rechteckige Topologie, so dass eine einfache, zusammenhängende Parametrisierung (d. h. das Urbild der Fläche in Parameterdarstellung) erfolgt. Im T-Netz werden die Kontrollpunkte der Fläche zusätzlich generisch hinterlegt, d. h. es erfolgt eine Trennung zwischen dem Parameterraum und dem Geometrieraum. Jeder Kontrollpunkt im T-Netz besitzt eine festes Parameterpaar als Position im T-Netz, ist in seiner geometrischen Lage im Raum allerdings frei verschiebbar. Somit modelliert das T-Netz das parametrische Urbild für die resultierende Freiformfläche.

Innere Kontrollpunkte in zweidimensionalen T-Netzen besitzen eine Valenz (d. h. Anzahl eingehender Kanten am Kontrollpunkt) von zwei (Eckpunkt, durchlaufender I-Punkt oder abknickender L-Punkt[3]), drei (T-Punkt) oder vier (Kreuzungspunkt, Abbildung 5.11). Äußere Kontrollpunkte als Eck- oder Randpunkte haben eine Valenz von zwei oder drei. Darüber hinaus sind die Kanten per Vereinbarung stets als Rechtecke zueinander ausgerichtet (siehe Abschnitt 5.1.2).

Falls ein T-Netz keine inneren T-Punkte enthält, dann repräsentiert es ein vollbesetztes Gitter und die resultierende T-Spline-Fläche entspricht einer NURBS-Fläche. Anders ausgedrückt kann ein T-Netz aus einem vorhandenen Kontrollpunktgitter entstehen, indem einzelne Kanten entfernt werden. Somit sind sämtliche topologische Formen für T-Netze möglich, bei denen Kanten aus einem Gitter entnommen werden (Abbildung 5.2).

5.4.2 Suchen und Nachbarschaftsbeziehungen

Ein Suchlauf über die Komponenten eines T-Netzes kann unsortiert erfolgen, d. h. das einfache Auflisten aller Typen (Punkt, Kante, Zelle), oder anhand einer strukturierten Suche in der Ebene des Parameterraums. Eine häufige Operation im Kontext der Knotenwertherleitung ist die Berechnung von Strahlschnitten auf einem T-Netz. Nicht

[3] Hier muss ggf. vereinbart werden, ob diese Formen für innere Kontrollpunkte angenommen werden.

zuletzt ist eine effiziente Identifikation von benachbarten Zellen notwendig, z. B. für die Verfeinerung eines T-Netzes.

5.4.3 Herleiten der Knotenwerte

Die Herleitung der Knotenwerte erfolgt wie im vorherigen Abschnitt 5.2.2 beschrieben. An dieser Stelle wird wiederholt, wie wichtig eine effiziente Schnittberechnung entlang der Parameterrichtungen im T-Netz ist.

5.4.4 Lokale Verfeinerung von T-Spline-Flächen

Eine entscheidende Operation für die Flächenrückführung stellt die lokale Verfeinerung von T-Spline-Flächen dar. Diese beruht auf der Unterteilung von Zellen im T-Netz [75].

Das Einfügen von neuen Kontrollpunkten in ein T-Netz ist bereits Teil der ursprünglichen Publikation von Sederberg et al. [77]. Ein Kontrollpunkt kann nur dann in ein bestehendes T-Netz eingefügt werden, wenn die T-Netz-Regeln (Abschnitt 5.2.3) nicht verletzt werden. Daraus ergibt sich, dass das Einfügen eines Kontrollpunkts in ein T-Netz nicht allgemein funktioniert, sondern wesentlich von der Netzstruktur abhängt. Neue Kontrollpunkte dürfen nur auf vorhandenen Kanten eingefügt werden, das Einfügen in einer Zelle darf wiederum nur nach einer vorherigen Unterteilung dieser Zelle durch eine Kante stattfinden. Schließlich wird dem neuen Kontrollpunkt eine rational summierte Geometriekoordinate der umliegenden Punkte zugewiesen.

Sederberg und Cardon überarbeiten den Verfeinerungsalgorithmus in [15] und [75]. Das Einfügen von neuen Kontrollpunkten ist somit allgemeingültig und mehrere Kontrollpunkte können gleichzeitig eingefügt werden. Der Algorithmus beruht auf einer sich wiederholenden Topologie- und einer finalen Geometriephase. Die Topologiephase fügt kaskadiert Kontrollpunkte ein, bis ein Regelsatz an möglichen Verletzungen zwischen den Knotenwerten und der Topologie erfolgreich abgearbeitet ist. Die Geometriephase aktualisiert die Geometriekoordinaten und die Gewichte der Kontrollpunkte. Während des Prozesses werden temporäre, invalide T-Netze erzeugt, d. h. es erfolgt eine Trennung der lokalen Knotenwerte von der aktuellen Netzrepräsentation.

Aus beiden Verfeinerungsalgorithmen ergibt sich die Notwendigkeit, die Zellen eines T-Netzes zu durchlaufen und auf diesen Suchanfragen im Parameterraum durchzuführen. Die Berechnung aktualisierter Knotenwerte ist eine zeitkritische Operation und wird im erweiterten Algorithmus sehr häufig benötigt. Bestehende Zellen im T-Netz müssen effizient unterteilt werden. Nicht zuletzt ist eine Datenstruktur wichtig, die eine (vorübergehende) Entkopplung der Topologie von den lokalen Knotenwerten und der Geometrie berücksichtigt.

5.4.5 Verbindung von T-Spline-Flächen

Sederberg et al. und Ipson beschreiben einen Algorithmus, der B-Spline-Flächen (bzw. ohne explizite Erwähnung beliebige T-Spline-Flächen) zu einer konsistenten T-Spline-Fläche mit einem Übergang frei wählbarer parametrischer Stetigkeit (Abschnitt 2.2.4) verbindet [77] [42]. Im Unterschied zu den üblichen Vorgehensweisen, um Spline-Flächen zu verbinden, werden die gitterförmigen Kontrollpunktnetze gegenseitig lokal verfeinert und anschließend unter Beachtung der Stetigkeit zusammengefügt. In einer weiteren Phase wird die Geometrie der Kontrollpunkte angepasst. Letztlich ergeben diese eine zusammenhängende T-Spline-Fläche mit einer frei wählbaren Stetigkeit

im Übergang (Abbildung 5.12). Die entstehenden T-Netze weichen i. A. von einer rechteckigen Topologie ab, da sie an beliebiger Stelle verbunden werden dürfen.

Die Verbindung von T-Spline-Flächen erfolgt auf der vorgestellten Datenstruktur mit einem vereinfachten Ansatz (Abschnitt 5.5.2).

5.4.6 Weitere Operationen

Die zusätzlich genannten Operationen und ihre Voraussetzungen sind für die Reduktion bathymetrischer Flächen und die im folgenden Abschnitt beschriebene Datenstruktur für T-Netze von untergeordneter Bedeutung. Sie stellen aber durchaus grundlegende Konzepte für weiterführende Arbeiten mit alternativen Schritten der Reduktion dar.

Vergröberung

Die „Vergröberung“ oder auch „Vereinfachung“ (*simplification*) genannte Operation beschreibt den umgekehrten Prozess der Verfeinerung. Aus einem anfangs hochaufgelösten T-Netz wird ein Kontrollpunktnetz für eine weitgehend „ähnliche“ Fläche mit reduzierter Anzahl an Kontrollpunkten erzeugt. Nach Sederberg et al. [75] [15] werden T-Spline-Flächen in Sequenzen $T_0 \subset T_1 \subset \ldots \subset T_n$ aus verschachtelten T-Spline-Räumen (*T-spline spaces*) eingeordnet. Ein T-Spline-Raum umfasst alle Flächen, die mittels eines fixen T-Netzes modelliert werden können. Gemeint ist, dass die Topologie des T-Netzes im Parameterraum vorgegeben ist, seine Geometriekoordinaten allerdings frei wählbar sind. Eine T-Spline-Fläche aus dem Raum T_{i-1} entsteht aus einem vereinfachten T-Netz zu einer T-Spline-Fläche aus T_i. Der Raum T_0 entspricht dem kleinstmöglichen T-Netz einer stark vereinfachten T-Spline-Fläche (z. B. ein einfaches Gitter). Die von den Autoren vorgestellte Vergröberung einer T-Spline-Fläche erzeugt in jedem Schritt ein T-Netz aus einem untergeordneten T-Spline-Raum und optimiert die Geometrie der Kontrollpunkte mithilfe der Methode der kleinsten Quadrate (Abschnitt 4.4).

Die Netzvergröberung kann als alternativer Top-Down-Ansatz für das vorgestellte Approximationsschema verwendet werden (Abschnitt 4.1). Hierfür wird zunächst ein sehr fein aufgelöstes T-Netz konstruiert, und dieses wird schrittweise durch Entfernen von Kanten vergröbert. Aus praktischer Sicht ist dieser Ansatz allerdings nicht durchführbar, da hochaufgelöste T-Netze als Eingangsdatensatz bereits einen hohen Rechen- und Speicheraufwand erfordern.

Null-Knotenintervalle

Knotenintervalle dürfen per Definition die Länge 0 annehmen, so dass Unstetigkeiten in der T-Spline-Fläche zugelassen sind. Die genaue Form der T-Spline-Fläche ergibt sich dann aus dem Grad. Nullintervalle, d. h. doppelte Knotenwerte, führen möglicherweise zu Unstetigkeiten innerhalb einer linearen T-Spline-Fläche (z. B. Knicke). Bei höheren Graden sind hierfür bereits mehrfach wiederholte Knotenwerte notwendig.

Da die Parameterwerte der Kontrollpunkte nicht mehr eindeutig sind, erfordert dies die Einbettung des T-Netzes in einen übergeordneten Raum mit diskreten Bildkoordinaten (Abschnitt 5.1.4). Ein Kontrollpunkt ist somit ein Tupel $(\mathbf{i}, \mathbf{p}, \mathbf{c}, w)$ aus zweidimensionalen Bild-, zweidimensionalen Parameter- und dreidimensionalen Geometriekoordinaten sowie dem Gewicht.

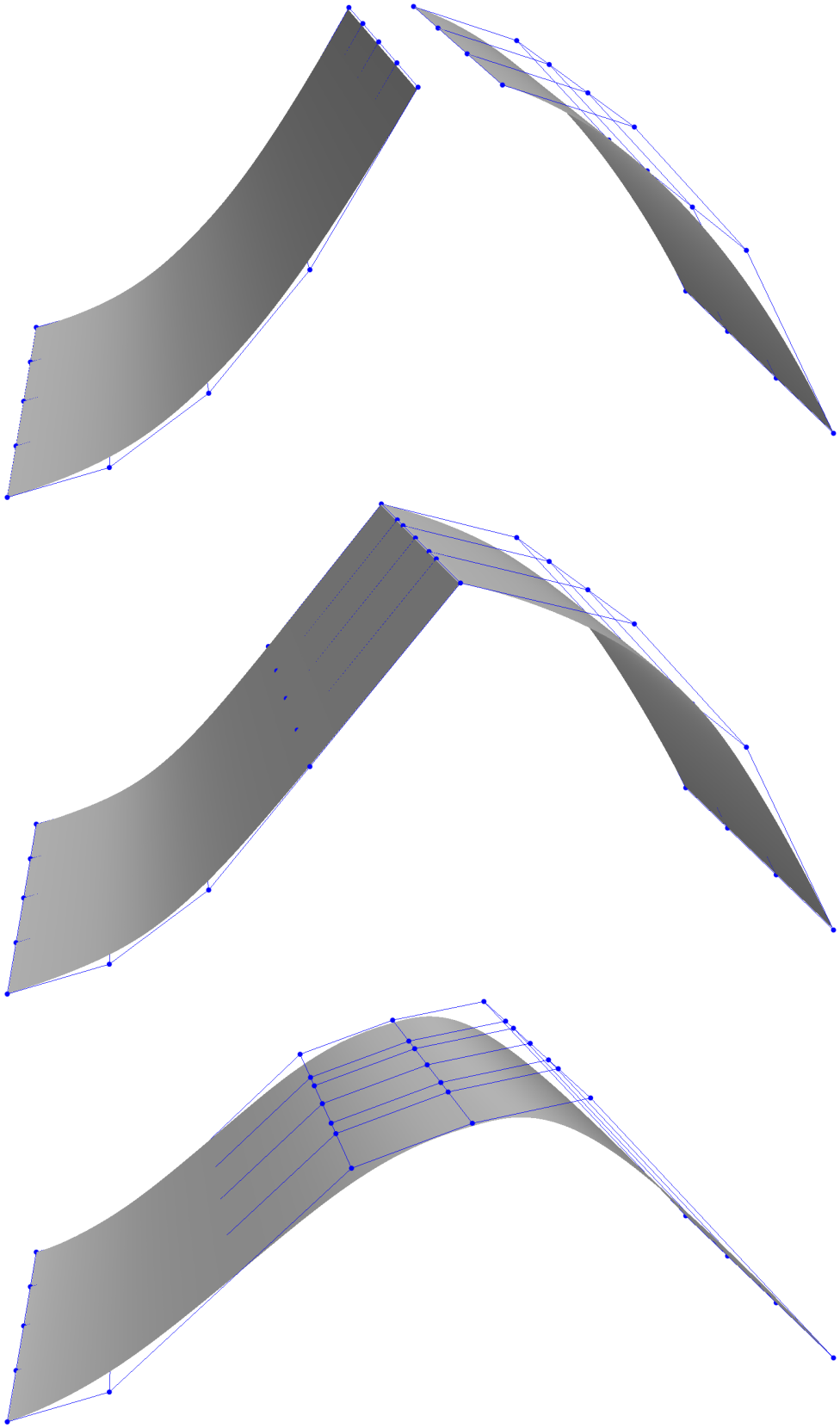


Abbildung 5.12: Verbindung von Flächen (oben), T-Spline-Fläche mit C^0-stetigem Übergang (Mitte) und C^2-stetigem Übergang (unten)

Extraktion von Bézier-Flächen

T-Spline-Flächen können in Bézier-Flächen konvertiert werden [77]. Dies kann global erfolgen oder auf lokale Bereiche beschränkt sein. Im Wesentlichen werden fehlende Kontrollpunkte mittels lokaler Verfeinerung sukzessiv eingefügt, bis ein vollständig besetztes Kontrollpunktgitter entsteht. Dieser Prozess ermöglicht die Konvertierung einer T-Spline-Fläche in Bézier- oder NURBS-Flächen, so dass diese in CAD-Systemen importiert werden können.

Höherdimensionale T-Zellkomplexe

Die im folgenden Abschnitt beschriebene Datenstruktur soll prinzipiell auf höhere Dimensionen erweiterbar sein. Hierfür bieten sich die gängigen Verfahren der Tensorprodukt-Methode an (z. B. dreidimensionale Quader anstelle von zweidimensionalen Rechtecken für Zellen im T-Netz), oder die Abbildung von separaten Teilgebieten einer Mannigfaltigkeit auf einzelne T-Spline-Flächen [36] [34].

5.5 Eine kombinierte Datenstruktur für T-Netze

Der vorgestellte Ansatz sieht eine Kombination aus einem starren Rechtecknetz mit einer allgemeinen, adaptiven Zellzerlegung vor. An dieser Stelle werden die Konstruktion einer solchen Datenstruktur motiviert und ihre Vorteile und Einschränkungen in Bezug auf die bathymetrische Flächenrückführung diskutiert.

5.5.1 Allgemeine R-Bäume

Ursprünglich gehen R-Bäume (für „*range tree*“) auf Guttmann zurück [35]. Ein solcher R-Baum ist eine dynamische, Baum-basierte Datenstruktur, die generische Objekte anhand räumlicher oder mehrdimensionaler Gebiete sortiert. Ein Objekt zeichnet sich, neben seinen individuellen Eigenschaften, insbesondere durch einen geometrischen Rahmenkörper (*bounding box*[4]) aus. Bauwerksteile besitzen beispielsweise konvexe Ausdehnungen, und hydrografische Vermessungen können durch ein Rechteck umfasst werden. R-Bäume haben gegenüber anderen multidimensionalen Datenstrukturen (z. B. Octrees / Quadtrees oder k-d-Bäume) Vorteile bzgl. ihrer, auf die Anzahl der Elemente bezogenen, logarithmischen Suchzeit.

Graphen und Bäume

Die umgesetzte Datenstruktur beruht auf den im Folgenden vorgestellten Grundlagen der Graphentheorie [9]:

Ein Graph ist ein Paar (V, E) bestehend aus zwei Mengen $V := \{v_i\}$ für die Ecken und $E := \{e_j\}$ als Kanten $e := (v, w)$ über den Ecken v und w.

Bei einem gerichteten Graphen wird zwischen den Kanten (v, w) und (w, v) unterschieden, d. h. ein ungerichteter Graph kennt nur bidirektionale Verknüpfungen zwischen den Ecken.

Zwei Ecken v_1 und v_2 heißen adjazent oder benachbart, sofern sie eine gemeinsame Kante (v_1, v_2) oder (v_2, v_1) besitzen.

Ein (gerichteter) Weg ist eine Abfolge von Ecken $(v_1, \ldots, v_n)$ zwischen zwei beliebigen Ecken v_1 und v_n.

[4] Ein achsenparalleles Rechteck, das eine Punktmenge minimal umfasst.

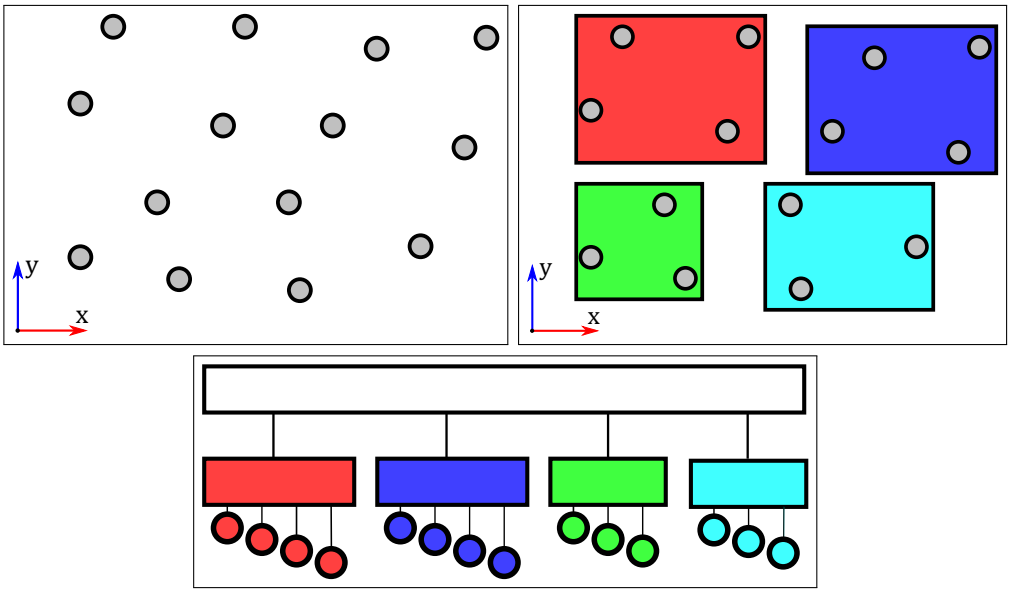


Abbildung 5.13: Darstellung einer Punktmenge (oben links), die Zuordnung in den R-Baum-Seiten (oben rechts) und der R-Baum (unten)

Liegt eine Ecke v_i auf einem Weg, so sind v_{i-1} der Vorgänger und v_{i+1} der Nachfolger der Ecke.

Ein Kreis ist ein Weg mit identischer Start- und Endecke. Ein Graph ist kreisfrei, sofern er keine Kreise enthält.

Ein spezieller Graphentyp sind Bäume: ein Baum ist ein gerichteter, kreisfreier Graph. Jede Ecke besitzt genau einen Vorgänger, bis auf die sogenannte Wurzel. Außerdem hat jede Ecke mindestens einen Nachfolger, andernfalls heißt sie Blatt. Ein k-närer Baum besitzt maximal k Nachfolger an jeder Ecke, die kein Blatt ist.

Datenstruktur

Ein Objekt in einem n-dimensionalen R-Baum wird durch ein Blatt verkörpert. Ein Blatt ist ein (Baum-)Knoten ohne Nachfolger. Jedes Blatt enthält einen Index-Eintrag auf eine separate Datenstruktur für die eigentlichen Objekte (z. B. Messpunkte oder Bauteile) und ein n-dimensionales Rechteck als Bounding-Box des Objekts. Die Objekte im R-Baum werden anschließend in Seiten (*pages*) zusammengefasst. Jede Seite umfasst eine maximal vorgegebene Anzahl k an Blättern und die vereinigte Bounding-Box aller enthaltenen Unterseiten. Ein Knoten des R-Baums besteht somit aus einer Seite, die wiederum maximal k Unterseiten enthält inkl. der vereinigten Bounding-Box. An oberster Stelle im R-Baum steht die Wurzel, die keine weiteren Vorgänger besitzt und deren Bounding-Box den gesamten Objektraum umschließt. Eine Darstellung von Punkten und die Einordnung in einem R-Baum mit maximal vier Elementen je Seite visualisiert Abbildung 5.13.

GPGPU-beschleunigte Implementierungen von R-Bäumen und deren Anwendungen mit Geodaten finden sich z. B. in [91] [68] [2] [96].

Suche in R-Bäumen

Räumliche Suchanfragen in Geoinformationssystemen werden oftmals mit R-Bäumen realisiert. Lokalisierungen, Überdeckungen oder Schnitte laufen stets von der Wurzel

aus. In jeder Seite wird zunächst anhand der Bounding-Box die jeweilige Anfrage getestet. Ist diese positiv, d. h. ein Punkt liegt z. B. in der größten Bounding-Box des R-Baums, so werden die Unterseiten im R-Baum sukzessiv durchlaufen und getestet. Findet sich jeweils eine Seite, so läuft der Algorithmus rekursiv weiter bis zum Erreichen eines Blattes. Das gefundene Blatt besitzt einen Index-Eintrag auf das zu suchende Objekt, so dass die Suche erfolgreich verläuft oder vorher abbricht. Ein letzter Test auf Schnitt o. ä. erfolgt auf dem gefundenen Objekt. Beispielsweise kann ein Punkt innerhalb einer Bounding-Box, allerdings außerhalb eines referenzierten Dreiecks liegen.

Füllen von R-Bäumen

Das effektive Lesen, Ändern und nachträgliche Beladen von R-Bäumen steigt und fällt mit der Verteilung der Elemente auf den Seiten. Ein R-Baum ist ungenügend austariert, falls seine Seiten in der Anzahl der nachfolgenden Unterseiten stark abweichen. Im schlechtesten Fall besitzt ein R-Baum stets nur eine Unterseite auf jeder Ebene. Der Baum nimmt dabei die Form einer verketteten Liste an, so dass die eigentlichen Suchanfragen maximal ineffizient verlaufen (lineare anstelle logarithmischer Laufzeit). Es ist wichtig, einen R-Baum bereits mit seiner Erzeugung ausgeglichen zu befüllen (sogenanntes *bulk loading*). Nachträgliche Änderungen (Einfügen, Löschen) erzwingen ebenfalls eine Neuverteilung der Objekte in den Baumknoten. Das Einfügen von Elementen findet deutlich effizienter statt, sofern der Baum anfänglich gut gefüllt wird und das Einfügen verteilter Elemente gleichmäßig erfolgt.

Die optimale Befüllungsstrategie ist anwendungsabhängig. Für Geodaten eignet sich oftmals eine räumliche Zusammenfassung der Messpunkte, so dass lokal korrelierende Messpunkte stets in einer Seite landen. Alternative Ansätze teilen die Daten anhand ihrer Verteilungsdichte oder schlicht anhand einer x-y-Sortierung der Geokoordinaten auf (s. u.). Das Ziel ist stets, die Daten auf alle Ebenen im R-Baum gleichmäßig zu verteilen.

5.5.2 Rechteck-R-Bäume

Rechteck-R-Bäume stellen eine Datenstruktur für T-Netze dar, die klassische Rechtecknetze um eine adaptive Zerlegung von Zellen ergänzen. Die Datenstruktur bietet den Vorteil, dynamische Operationen wie das Einfügen oder Löschen von Zellen im T-Netz zu unterstützen und gleichzeitig räumliche Suchanfragen effizient zu beantworten. Aufgrund der reduzierten Datenstruktur ist die Bestimmung von Zellen in Bezug auf Nachbarschaften und Inklusion ebenso recht einfach.

An die T-Netze wird die Forderung gestellt, dass jede Zelle im T-Netz genau vier Kontrollpunkte, jeweils mit einem Tupel aus dem Parameterpaar, der Geometrie und dem Gewicht, enthält. Dies schließt z. B. T-Netze mit I-Punkten aus (siehe Abschnitt 5.4.1), bei denen Kontrollpunkte einzeln auf einer Kante liegen. T-Knoten sind allerdings erlaubt, da derartige Kreuzungen stets Eckpunkte von zwei Zellen sind (Abbildung 5.14)

Jede T-Netz-Zelle ist durch ein achsenparalleles Rechteck beschrieben, d. h. ein Paar von minimalen und maximalen Parameterpaaren $(s_{\min}, t_{\min})$ und $(s_{\max}, t_{\max})$. Offensichtlich entspricht die Bounding-Box einer Zelle ihren tatsächlichen Dimensionen im Parameterraum. Die zweidimensionale Strukturierung der T-Netz-Zellen erfolgt nun durch einen R-Baum. Aus jeder Zelle im T-Netz werden die vier Kontrollpunkte der Ecken entnommen. Ihre Parameterwerte geben die Bounding-Box im R-Baum vor.

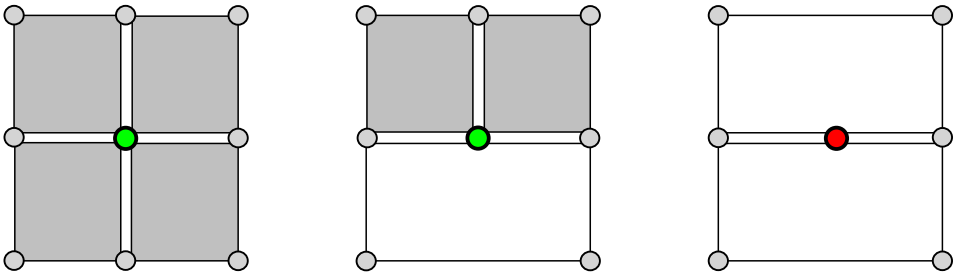


Abbildung 5.14: Akzeptierte Kontrollpunkte im R-Baum: Die grünen Kontrollpunkte sind gültig, da sie Eckpunkte von Zellen (grau) sind. Innenliegende Kontrollpunkte sind ungültig (rot).

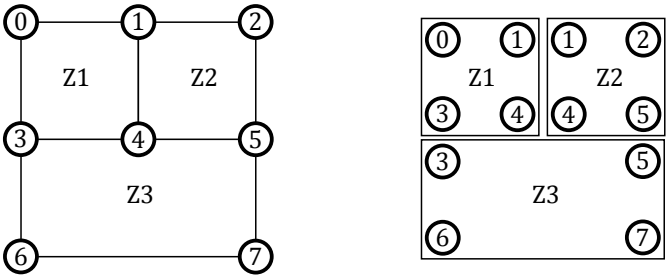


Abbildung 5.15: Beispielhaftes T-Netz (links) und die dazugehörige Zerlegung der Zellen $Z1$, $Z2$ und $Z3$ im R-Baum (rechts)

Die Blätter des R-Baums verweisen auf die ursprünglichen T-Netz-Zellen mittels einer separaten Indexstruktur (Abbildung 5.15). Die Tabelle 5.1 bis 5.3 geben die Belegung der Datenstrukturen für einen R-Baum mit jeweils zwei Elementen pro Seite vor. Zu beachten ist das mehrfache Auftreten von Kontrollpunkten in der Punktliste der Zellen. Eine Ergänzung wäre eine zusätzliche Referenzierungstabelle für die Abbildung der Eckpunkte zu den Kontrollpunkten des T-Netzes, die allerdings mit größerem Speicherbedarf und einem zusätzlichen Nachschlagen verbunden ist.

Die Datenstruktur findet Einsatz in der Reduktion von bathymetrischen Flächen. In Abschnitt 4.1.1 wird ein einfaches T-Netz als Basisnetz für das Approximationsschema erzeugt. Bei einem initialen T-Netz (z. B. 3×3 Kontrollpunkte) ergibt dies eine Menge an Rechtecken (2×2). Die erzeugten T-Netze werden ab diesem Zeitpunkt stets durch lokale Verfeinerungen verändert. Ferner ergeben sich hieraus Redundanzen, da

Punkt	**s**	**t**
P_0	0	2
P_1	1	2
P_2	2	2
P_3	0	1
P_4	1	1
P_5	2	1
P_6	0	0
P_7	2	0

Tabelle 5.1: Parameterpaare der Kontrollpunkte

Seite	Typ	Bounding-Box	Unterseiten	Zelle
0	Wurzel	(0,0) - (2,2)	1,2	-
1	Unterseite	(0,1) - (2,2)	3,4	-
2	Blatt	(0,0) - (2,1)	-	Z3
3	Blatt	(1,1) - (2,1)	-	Z2
4	Blatt	(0,1) - (1,2)	-	Z1

Tabelle 5.2: Belegung des R-Baums

Zelle	Punkte
Z1	(P_0,P_1,P_3,P_4)
Z2	(P_1,P_2,P_4,P_5)
Z3	(P_3,P_5,P_6,P_7)

Tabelle 5.3: Punktliste der Zellen im T-Netz

Kontrollpunkte und Kanten innerhalb der Datenstruktur mehrfach erfasst sind (z. B. durch T-Kreuzungen, Details s. u.).

In den folgenden Abschnitten werden die benötigten Eigenschaften und Anpassungen der Operationen aus Abschnitt 5.4 beschrieben. Eine OpenCL-Implementierung der vorgestellten Datenstruktur ist in Abschnitt 6.2 gegeben.

Füllstrategie

„*Nearest X*“ bezeichnet eine effektive und schnelle Füllstrategie, die zunächst eine Ordnungsrelation für ein aufsteigendes Sortieren der Objekte in x-y-Reihenfolge erfordert. Jede Zelle in einem T-Netz besitzt einen parametrischen Mittelpunkt, so dass dieser als Referenz für die Sortierung herangezogen wird. Daran anschließend verteilt die Strategie die einzelnen T-Netz-Zellen auf die Seiten im R-Baum bis zu der maximal vorgegebenen Anzahl k an Objekten in den Blättern.

Im Anschluss werden die Blätter zu Seiten mit maximal k Unterseiten aggregiert. Dieses Vorgehen wird wiederholt, bis zuletzt eine (Wurzel-)Seite mit maximal k Unterseiten entsteht.

Abbildung 5.16 zeigt eine beispielhafte Verteilung der Zellen gemäß der Nearest-X-Füllstrategie.

Kombination der Topologie und Geometrie

Grundsätzlich kombinieren Rechteck-R-Bäume die Topologie des T-Netzes und die Geometrie der T-Spline-Fläche. Die Blätter im R-Baum referenzieren die Zellen des T-Netzes, und diese enthalten die Kontrollpunkte samt Geometrie. Das parametrische Definitionsgebiet der T-Spline-Fläche ergibt sich über die Vereinigung der parametrischen Bounding-Boxen in den Blättern des R-Baums. Die Datenstruktur bietet darüber hinaus genügend Flexibilität, um Löcher oder zusammenhangslose Teilgebiete eines T-Netzes zu erfassen.

Lokale Verfeinerung

Die lokale Verfeinerung in einem Rechteck-R-Baum weicht von der allgemeinen Vorgehensweise der lokalen Verfeinerung in T-Spline-Flächen ab (Abschnitt 5.4.4). Anstelle

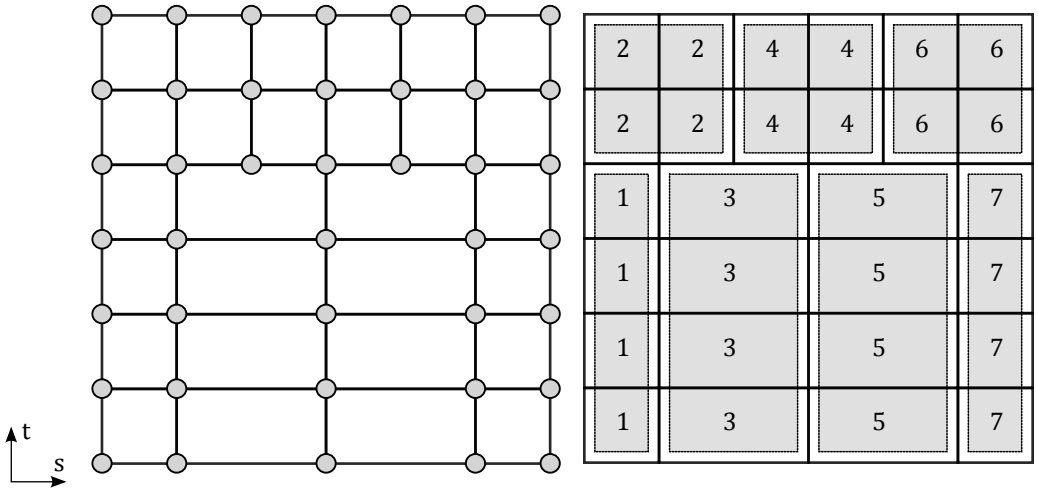


Abbildung 5.16: Zellen im T-Netz (links) und ihre Zuordnung zu den Seiten nach der Nearest-X-Strategie (rechts)

des Einfügens von neuen Kontrollpunkten und Knotenwerten, werden direkt Zellen unterteilt und somit indirekt neue Kontrollpunkte im T-Netz eingefügt.

Ist die zu unterteilende Zelle bekannt (z. B. über eine Suchanfrage ermittelt), so werden vier neue Unterzellen im anteiligen Verhältnis erzeugt. Dies kann zentriert erfolgen oder in einem Unterteilungsverhältnis, das dem Parameterpaar des gewünschten neuen Kontrollpunkts entspricht. Die neuen Zellen werden direkt in den Rechteck-R-Baum eingefügt. Zu beachten ist, dass die alte, nicht-unterteilte Zelle weiterhin im R-Baum existiert. Diese wird ggf. für eine nachfolgende Vergröberung benötigt. Der Prozess erzeugt insgesamt fünf neue Kontrollpunkte (oben, unten, links, rechts, zentral), denen noch keine geometrischen Koordinaten zugewiesen sind. Diese können sofort aus der bestehenden Geometrie interpoliert werden, oder sie werden erst zum Zeitpunkt der Flächenrückführung nachträglich bestimmt (Abschnitt 4.3).

Da die alte Zelle beibehalten wird, gilt es zu beachten, dass Suchanfragen von nun an stets die Historie der Zellunterteilung ermitteln. Genaueres hierzu wird in der Suche beschrieben (s. u.).

Vergröberung

Umgekehrt können zusammenliegende Zellen im T-Netz vergröbert werden, sofern sie aus einer vorherigen lokalen Verfeinerung heraus entstanden sind. Ist eine Zelle ausgewählt, deren Unterteilung aufgehoben werden soll, so ergibt eine Suchanfrage nach ihrer Bounding-Box alle in ihr enthaltenen Kind-Zellen, die direkt aus dem R-Baum gelöscht werden.

Suche und Nachbarschaftsbeziehungen

Die Lokalisierung eines Punkts („Liegt Punkt P im Parameterraum des T-Netzes?“) erfolgt durch direkte Suchanfragen im R-Baum. Es können punktuelle, linienhafte (für die Berechnung von Schnittstrahlen, s. u.) oder flächenhafte Anfragen (für die Bestimmung von Teilflächen) gestellt werden.

Suchanfragen im R-Baum erlauben die Bestimmung von Nachbarschaftsbeziehungen zwischen den Zellen des T-Netzes. Sind zwei Zellen im T-Netz benachbart, so gibt es eine gemeinsame (Teil-)Kante als Schnittmenge. Gesucht werden somit genau die

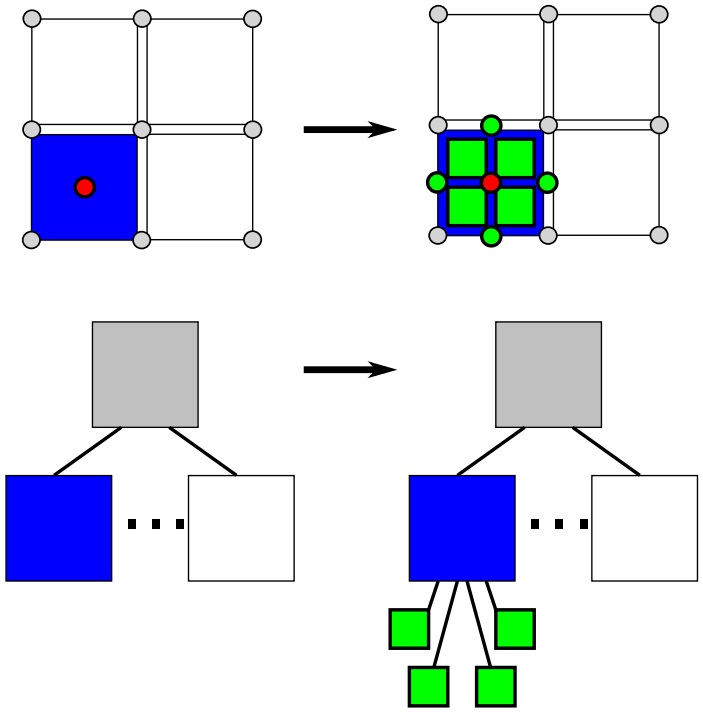


Abbildung 5.17: Oben: Verfeinerung der T-Netz-Zellen am Mittelpunkt (rot) und die neuen Kontrollpunkte (grün). Unten: Einfügen der neu erzeugten Objekte im R-Baum unterhalb der bestehenden Seite.

Nachbarzellen, die eine nichtleere Überdeckung mit der Bounding-Box der aktuellen Zelle besitzen.

Aus der Menge der Nachbarn kann nun gezielt nach Himmelsrichtungen ausgewählt werden, indem diese anhand ihrer Koordinaten verglichen werden (westlich gelegene Zellen haben z. B. immer kleinere oder gleiche vertikale Parameterwerte).

Sofern das betrachtete T-Netz in seinem aktuellen Zustand eine Folge von Verfeinerungen ist, entstehen Hierarchien im R-Baum. Somit wechseln Blätter zu Seiten im R-Baum und decken im veränderten Zustand weitere Unterzellen ab. Eine Suche im Parameterraum erfasst sämtliche dieser Zellen. Dabei ist zu unterscheiden, ob lediglich Kontrollpunkte benötigt werden (diese werden in der separaten Liste ohnehin nur einmalig erfasst), oder ob Schnittpunkte auf Kanten ermittelt werden. Letzteres führt zu redundanten Schnittberechnungen.

Soll die Suche nach Kontrollpunkten optimiert werden, so können diese in einem separaten R-Baum gespeichert werden. Die Blätter des Baums verweisen dann direkt auf die Kontrollpunkte anstelle der T-Netz-Zellen. Eine auf diesem Gedanken beruhende OpenCL-Implementierung für PB-Spline-Flächen folgt in Abschnitt 6.2.2.

Knotenwert-Herleitung

In Abschnitt 5.2.2 ist die Herleitung der lokalen Knotenwerte bei ungeradem Grad beschrieben. Hierzu werden, ausgehend von einem Kontrollpunkt, wiederholt Schnittberechnungen im T-Netz durchgeführt, um die Parameterwerte der benachbarten Zellen zu ermitteln. Die in dieser Arbeit eingeführten Rechteck-R-Bäume sind somit eine auf die Beantwortung von Suchanfragen optimierte Datenstruktur für die Schnittberechnung bei der Knotenherleitung. Im Kern reduziert sich diese Operation auf eine

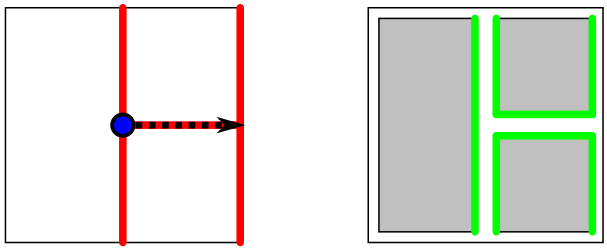


Abbildung 5.18: Darstellung des Schnittstrahls entlang einer Kante mit geschnittenen Kanten (links) und die dadurch mehrfach geschnittenen Kanten (grün) und Zellen im R-Baum (rechts)

linienhafte Suche (s. o.). Aus der Menge an geschnittenen Zellen im T-Netz werden in einem weiteren Schritt die tatsächlich geschnittenen Kanten bestimmt.

An dieser Stelle ist zu beachten, dass die Schnittberechnungen stets mehrfache Ergebnisse liefern, da Kanten zwischen benachbarten Zellen und nach Verfeinerungen mehrfach aufgefunden werden (Abbildung 5.18). In der vorliegenden Arbeit werden die mehrfach erfassten Knotenwerte aus dem Suchergebnis des Schnitts entfernt. Dies garantiert, dass keine parametrischen Unstetigkeiten in der T-Spline-Fläche auftreten.

Verbindung

Die Verbindung von T-Netzen erfolgt durch eine Vereinigung der Rechteck-R-Bäume. Es entsteht ein loser, nicht unbedingt zusammenhängender, Verbund aus Zellen. Die Herleitung der Knotenwerte findet ebenfalls direkt auf der vereinigten Menge der Zellen statt, da die Schnittberechnungen hier keine Änderungen benötigen. Der Prozess kann sukzessiv mit weiteren T-Netzen durchgeführt werden.

Zu beachten ist, dass die resultierende T-Spline-Fläche nur eine C^0-Stetigkeit im Übergang garantiert (im Vergleich mit Ipson und Sederberg et al. [42] [77]). Da ohnehin eine abschließende Korrektur der Geometrie durch die approximative Flächenrückführung erfolgt, wird das Problem in den Hintergrund gestellt.

Export eines T-Netzes

Ein Netzexport berücksichtigt lediglich die Blätter des R-Baums. Diese enthalten die Informationen über die eigentlichen Zellen des T-Netzes und die Verweise auf die Kontrollpunkte. Für die Berechnung einer T-Spline-Fläche genügen hingegen die Kontrollpunkte und die lokalen Knotenwerte. Die hierfür benötigten Informationen liegen vollständig im Rechteck-R-Baum.

Das Einfügen von weiteren Kontrollpunkten (d. h. lokale Verfeinerungen) ermöglicht die Konstruktion eines vollständigen Kontrollpunktgitters in Teilen von oder im gesamten T-Netz. Dieses Vorgehen unterstützt die Extraktion von (Teil-)Bézier-Flächen der T-Spline-Fläche (Abschnitt 5.4.6).

5.6 Zwischenstand

T-Netze sind von immanenter Bedeutung für T-Spline-Flächen. Eine effiziente Umsetzung der Kontrollpunktnetze ermöglicht erst das effektive Arbeiten mit T-Spline-Flächen. Das vorgestellte Approximationsschema basiert auf der simultanen Verfeinerung des Netzes, da in jeder Iteration eine Menge an neuen Kontrollpunkten

bestimmt und eingefügt wird. Anschließend erfolgt die Herleitung der neuen Knotenwerte. Die wiederholt durchgeführte Kombination der beiden Operationen motiviert eine massiv-parallele Beschleunigung. Im nachfolgenden Kapitel wird eine OpenCL-Implementierung vorgestellt, die auf diese Erkenntnisse zurückgreift.

Von Nachteil an der vorgestellten Datenstruktur ist, dass stets eine vollständige Hierarchie an Zellen abgelegt wird. Die Blätter des R-Baums enthalten Verweise auf die Zellen im T-Netz, weiterhin belegen die Bounding-Boxen im Baum zusätzlichen Speicherplatz. Unter Berücksichtigung der positiven Aspekte ist diese Redundanz zu vernachlässigen. Letztlich arbeiten die Suchstrategien insgesamt sehr effektiv, da sie auf der logarithmischen Suchtiefe der R-Bäume beruhen.

6 GPGPU-Optimierung

In den dargelegten Schritten zur Reduktion bathymetrischer Flächen kristallisieren sich zeitkritische und rechenintensive Aufgaben heraus. Die Verwendung der GPGPU-Schnittstelle OpenCL schafft die Voraussetzung für eine effektive massiv-parallele Berechnung. Wesentliche Bestandteile der Berechnungen benötigen allerdings hierfür angepasste Datenstrukturen und Algorithmen. In diesem Zusammenhang fließen nun neuartige Ansätze für die OpenCL-Integration in Teilschritte der Flächenrückführung und der bathymetrischen Geländemodellierung ein.

6.1 OpenCL Architektur

Im Folgenden werden die für diese Arbeit relevanten Bestandteile der virtuellen Architektur von OpenCL vorgestellt. Die Darstellung der wesentlichen Merkmale ist für die Reduktion von Bathymetrien insofern von Bedeutung, da die umgesetzten parallelen Optimierungen sehr konkreten Bezug auf die OpenCL-Modelle nehmen.

6.1.1 Platform Model

Ein wesentlicher Vorteil in Form der Portabilität von OpenCL zeigt sich letztlich darin, dass in einem konkreten Szenario die Zielplattform der Hardware unbekannt ist. Anstelle dessen wird eine abstrahierte, virtuelle Hardware-Plattform beschrieben, die im Wesentlichen einen Kompromiss aus den gängigen Grafikkarten-Architekturen darstellt (Abbildung 6.1). Sie bildet die Kommunikation zwischen dem Gastsystem (*host*) als Umgebung des ausführenden Systems und dem eigentlichen Gerät (*device*) der GPGPU-Beschleunigung ab.

Sowohl Host als auch Device verfügen über eigene Recheneinheiten und einen unabhängigen, getrennten Speicherraum. Dem Host unterliegt die Aufgabe, einen OpenCL-*Context* für jedes Device zu arrangieren (es dürfen mehrere Devices vorhanden sein). Dies beinhaltet die Initialisierung, die Verwaltung zur Laufzeit und das finale Schließen des Context. Der Context stellt Möglichkeiten zur Kommandoverarbeitung im Device zur Verfügung. Dies geschieht über die *Command Queue*, die eine Warteschlange für die Zugriffe auf das Device bereitstellt. Operationen auf dem Device werden seriell durchgeführt und unterscheiden sich durch schreibende (Datentransfer vom Host zum Device), lesende (Datentransfer vom Device zum Host) und ausführende Zugriffe (Aufruf einer Kernel-Berechnung).

Die kleinste Einheit auf dem Device bilden die *Processing Elements* als atomarer (Rechen-)Knoten. Mehrere Processing Elements bilden einen Block, die sogenannte *Compute Unit*. Interne Synchronisation der Berechnungen findet lediglich innerhalb der Compute Units statt, d. h. es gibt keine weiteren speicherübergreifenden Datentransfers. Die genaue Anzahl an Processing Elements und Compute Units ist geräteabhängig. Der Grad maximaler Parallelität wird insgesamt durch die Gesamtzahl der Processing Elements bestimmt. Liegt die Anzahl der Aufrufe über der Anzahl der Processing Elements, so werden die Aufrufe blockweise hintereinander ausgeführt.

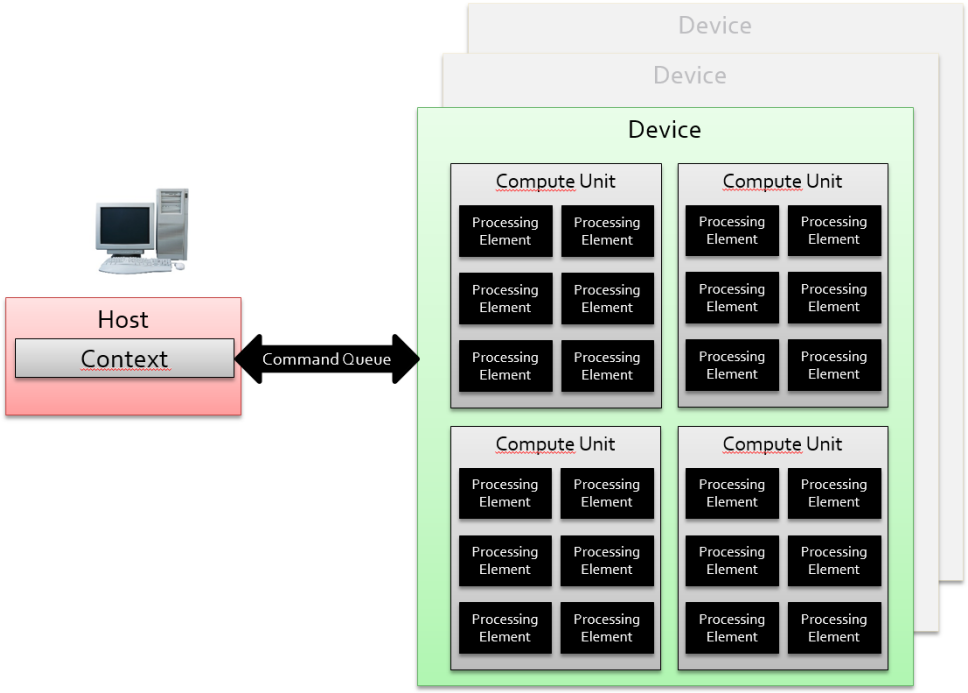


Abbildung 6.1: OpenCL Plattformmodell

6.1.2 Execution Model

OpenCL unterscheidet zwischen Host- und *Kernel*-Programmen. Host-Programme laufen in der Host-Umgebung (meist in den Programmiersprachen C++ oder Java geschrieben). Ein Kernel ist ein C-Programm, das in den Processing Elements ausgeführt wird. Die simultan aufgerufenen Instanzen des gleichen Kernels in den verfügbaren Processing Elements, bezeichnet als *Work Item*, ermöglichen erst die massive Parallelität im Ablauf eines Algorithmus. Work Items werden in *Work Groups* organisiert. Es ergeben sich 1-, 2- oder 3-dimensionale Strukturierungen der Work Items, konzeptbedingt bezeichnet als *NDRange*. Ein praktisches Beispiel ist die Multiplikation einer $M \times N$-Matrix, da hier die Work Items ebenfalls als zweidimensionale Matrix arrangiert sind. Jedes Work Item berechnet somit ein Element der Ergebnismatrix. Es ist eine interne Aufgabe der Work Items, anhand einer globalen Indizierung den Teil der Ein- oder Ausgabe zu erkennen, für den der aktuelle Aufruf gerade verantwortlich ist.

6.1.3 Memory Model

Das Speichermodell von OpenCL bildet einen Kompromiss zwischen den bekannten CPU- und GPU-basierten Speicherhierarchien. Es modelliert eine abstrakte Hierarchie, die auf die konkrete Zielplattform bestmöglich abgebildet wird (Abbildung 6.2). Dem Kernel stehen dabei folgende Speicherbereiche zur Verfügung (Details s. u.):

- *Global Memory*
- *Constant Memory*
- *Local Memory*
- *Private Memory*

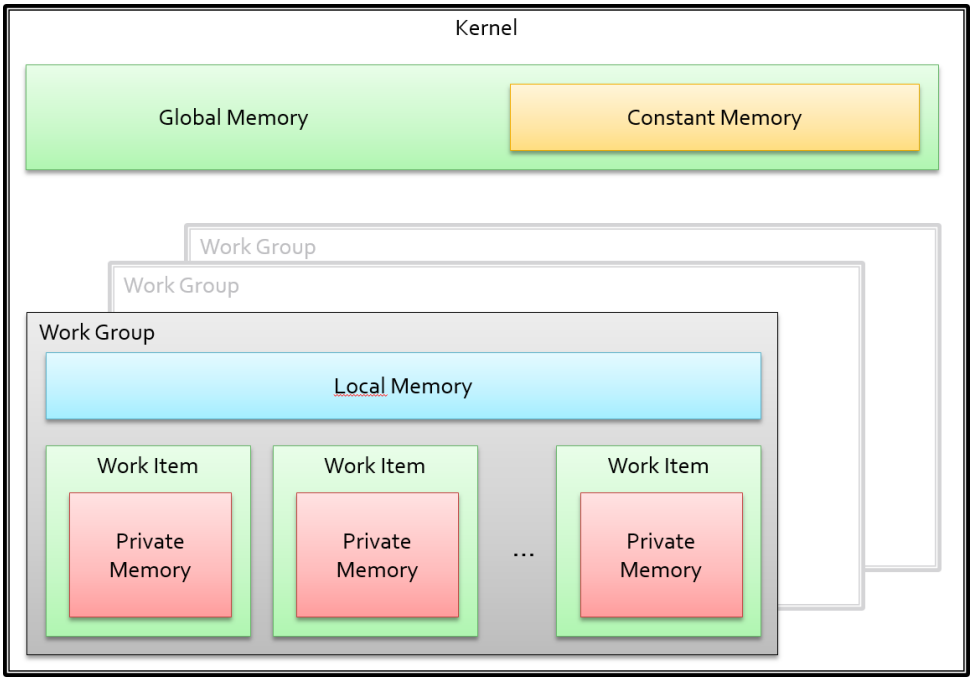


Abbildung 6.2: OpenCL Speichermodell

Darüber hinaus definiert OpenCL den *Host Memory* als den Speicher, der ausschließlich auf dem Host zur Verfügung steht. Über die Command Queue können Daten vom Host-Speicher auf den Device-Speicher transferiert werden.

Je nach Zielarchitektur können Speicherbereiche nur in geringer Größe oder gar nicht vorhanden sein. Es ist Aufgabe der jeweiligen Plattform, die Speicherbereiche somit transparent zu emulieren. Es empfiehlt sich ohnehin, OpenCL-Programme auf die konkret eingesetzte Plattform hin zu optimieren, d. h. die Dimensionierung und Auswahl des Speichers hardwarenah anzupassen.

Global Memory

Auf den globalen Speicher können alle Kernel-Programme lesend und schreibend zugreifen. Einerseits sind Zugriffe auf diesen Speicher mit hohen Latenzen und (beim Schreiben) Synchronisierungsproblemen verbunden, andererseits steht hier der größte Platz zur Verfügung. Der häufigste Gebrauch für globalen, beschreibbaren Speicher besteht in der Reservierung für die Ausgabedaten. Variablen im globalen Speicher erhalten die Kennzeichnung (*address space qualifier*) `__global`.

Constant Memory

Mit dem konstanten Speicher steht ein Nur-Lese-Speicher als Teil des globalen Speichers zur Verfügung. Er kann aus Leistungsgründen verwendet werden, da zum Zeitpunkt der Kompilierung garantiert wird, dass keine schreibenden und daher synchronisierten Zugriffe stattfinden. Unveränderliche Eingabedaten eines Algorithmus können hier abgelegt werden. Die Kennzeichnung lautet `__constant`.

Local Memory

Jede Work Group verfügt über einen gemeinsam genutzten Speicherraum. Die Lokalität bezieht sich hierbei auf die synchronisierten Zugriffe innerhalb der Work Group. Dies führt zu einem deutlich geringeren Synchronisationsaufwand. Die Kennzeichnung lautet `__local`.

Private Memory

Die kleinste Stufe in der Speicherhierarchie ist der private, individuelle Speicher jedes Work Item. Typischerweise werden hier die internen Variablen der Kernel-Instanzen abgelegt, z. B. Schleifenzähler, Zwischenergebnisse usw. Die optionale Kennzeichnung lautet `__private`.

6.1.4 Programming Model

Das Programmierungsmodell von OpenCL berücksichtigt explizite Vorgaben und einige implizite, nicht offensichtliche Einschränkungen bezogen auf die Programmierung. Einige der bereits genannten Aspekte fallen hierunter:

- **Datenparallelität** bezieht sich auf die Verarbeitung von Vektordaten mit höchstmöglicher Optimierung, z. B. die simultane Addition von Vektorkomponenten.

- **Task-Parallelität** ermöglicht die Auslastung von OpenCL-Devices ohne Beachtung der Verfügbarkeit. Allerdings ist die Reihenfolge der Aufrufe nicht garantiert.

- Work Items sind stets in n-dimensionalen Gruppen als **NDRange** arrangiert. Die Kernel-Programme sind immer im Kontext einer Gruppe zu betrachten.

- Synchronisation kann nur innerhalb einer Work Group stattfinden. Die realen Ausführungszeiten der atomaren Work Items sind nicht garantiert, daher definieren **Barrieren** explizite Zeitpunkte für eine Synchronisierung. Der Programmablauf wird solange angehalten, bis sämtliche Work Items die Barriere erreichen.

6.1.5 Programmiersprache OpenCL C

C99 ist eine ISO-zertifizierte Erweiterung der Programmiersprache C. Als hardwarenahe Sprache geht C99 als *OpenCL C* in die OpenCL-Architektur ein. Im Wesentlichen ist der Sprachumfang, d. h. Syntax und Datentypen, identisch und wird um eine API für Vektorbefehle erweitert. Tabelle 6.1 listet die hier relevanten Datentypen auf.

Vektordatentypen sind Erweiterungen skalarer Variablen zu einem multidimensionalen Vektor. Die beschriebenen Datentypen sind als Vektoren mit 1, 2, 4, 8 oder 16 Elementen verfügbar, als Schreibweise `xxxn`. So repräsentiert der Datentyp `int4` einen Vektor aus vier Ganzzahlen. Anweisungen auf einem Vektordatentyp werden gebündelt durchgeführt. Im vorherigen Beispiel können somit zwei Variablen vom Typ `int4` direkt addiert und zu einer neuen Vektorvariable gleichen Typs zusammengefasst werden. OpenCL definiert diverse Vektoroperationen, um diese miteinander zu verknüpfen.

Datentyp	Verwendung
`bool`	Boolean-Datentyp
`char`, `uchar`	Charakter-Datentyp (mit / ohne Vorzeichen)
`short`, `ushort`	Short-Datentyp (mit / ohne Vorzeichen)
`int`, `uint`	Integer-Datentyp (mit / ohne Vorzeichen)
`long`, `ulong`	Long-Datentyp (mit / ohne Vorzeichen)
`half`, `float`, `double`	IEE754-Gleitkommazahl mit halber / normaler / doppelter Genauigkeit (nicht alle Plattformen)
`image2d`, `image3d`	Raster- bzw. Voxeldatensatz mit wählbaren Farbkanaltypen und -sortierungen
`sampler_t`	Interpolation auf `image`-Datentypen (linear oder „Nächster Nachbar“)

Tabelle 6.1: Ausgewählte Datentypen in OpenCL nach [72] [33]

6.2 Optimierte Datenstrukturen für T-Netze

In diesem Abschnitt werden optimierte Datenstrukturen beschrieben, die für T-Netze eingesetzt werden. Zunächst wird eine abstrakte Kodierung für beliebige R-Bäume vorgestellt (d. h. losgelöst von der bathymetrischen Modellierung), anschließend folgen die konkreten Implementierungen für die abzubildenden T-Netze in OpenCL C.

6.2.1 OpenCL-Kodierung von generischen R-Bäumen

Ein R-Baum besteht aus einer Menge an Seiten und Blättern. Seiten referenzieren ihre Unterseiten und Blätter verweisen auf eine externe, generische Datenliste (Abschnitt 5.5.1). In den hier umgesetzten zweidimensionalen R-Bäumen besitzt jede Seite eine Bounding-Box für die ganzzahlig gerundeten Intervalle des umfassenden Raums in den Unterseiten. Diese Intervalle sind somit vom gleichen Datentyp (*integer*), wie die Indizes der Unterseiten und werden als kombinierter Vektordatentyp abgelegt.

Die Suche in einem R-Baum ist eine häufig wiederkehrende Subroutine in allen folgenden Anwendungen. Die konkrete OpenCL-Implementierung ist jeweils abhängig vom Szenario, daher wird die generische Suche angepasst und in den jeweiligen Rechenschritt eingebettet. So wird in den Kernel-Programmen wahlweise ein Kontrollpunkt (für die Berechnung von PB- bzw. T-Spline-Flächen) oder eine Zelle aus einem T-Netz gesucht (letztes erfolgt z. B. in den Schnittberechnungen). Algorithmus 5 stellt die generische Objektsuche in einem R-Baum als Pseudo-Code dar. Konkrete Details zu der OpenCL-Implementierung folgen im Anschluss.
Folgende Ergänzungen zu Algorithmus 5 sind zunächst nötig:

- Die Eingabe besteht aus der Suchkoordinate `p` im Parameterraum und einem Feld aus Seiten P (*pages*). `r` ist der Index der Wurzel.
- Eine *Queue* bezeichnet eine Warteschlangen-Datenstruktur, und besitzt somit Funktionen zum Einfügen an das Ende (`enqueue`) und Entfernen vom Anfang (`dequeue`).

```
Eingabe:
– Parameterpunkt p
– Seiten P (pages) im R-Baum
– Wurzel r (root) des R-Baums
begin
    – Initialisiere Ausgabeliste OUT := ∅
    – Initialisiere Queue Q := {r}
    repeat
        – Node n := Q.dequeue()
        if n ist Blatt then
            – OUT.add(n.object)
        else
            if p liegt in n.rectangle then
                foreach Unterseite c von n do
                    – Q.enqueue(c)
                end
            end
        end
    until Q leer;
end
Ausgabe: OUT
```

Algorithmus 5 : Generische Suche im R-Baum als Pseudocode

- `n.rectangle` ist die Bounding-Box der Seite. Der Test „liegt in“ erfolgt über Koordinatenvergleiche.
- `n.object` ist der externe Index des generischen Objekts im R-Baum.
- Die Ausgabe `OUT` beinhaltet alle gefundenen generischen Objekte. Ggf. wird bereits nach dem ersten Ergebnis abgebrochen. Wichtig ist dabei, dass es sich um Kandidaten der generischen Objekte handelt. Die tatsächliche Suche muss die Kandidaten einzeln aufgreifen und ggf. testen. Beispielsweise besitzen Dreiecke zwar eine Bounding-Box, allerdings kann ein Punkt weiterhin außerhalb des Dreiecks liegen.
- Der Suchaufwand ergibt sich aus der Tiefe des Baums. Ein ausgewogener R-Baum erreicht logarithmische Zugriffszeiten bezogen auf die Anzahl seiner Objekte.

Die Suche nach einem Datenobjekt wird als Unterroutine in allen nachfolgenden Operationen eingesetzt werden. Hierfür werden konkrete Anpassungen vorgenommen und diese in OpenCL C implementiert. Es ist möglich, eine generische Seite für allgemeine zweidimensionale Koordinaten vollständig innerhalb eines ganzzahligen `int8`-Vektors zu kodieren:

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
`int8`							
$\lfloor x_{\min} \rfloor$	$\lfloor y_{\min} \rfloor$	$\lceil x_{\max} \rceil$	$\lceil y_{\max} \rceil$	c_0	c_1	c_2	i_{obj}

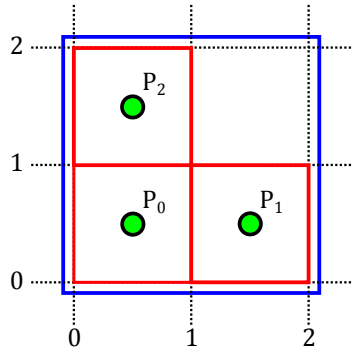


Abbildung 6.3: Punkte und Bounding-Boxen des R-Baums (rote Box für Punkte, blaue Box für Vereinigung)

Index	x	y
0	0.5	0.5
1	1.5	0.5
2	0.5	1.5

Tabelle 6.2: Punktkoordinaten zu Abbildung 6.3

Eine Seite im R-Baum umfasst dabei

- die vier ganzzahligen Koordinaten der Bounding-Box (ab- bzw. aufgerundet).
- die positiven Indizes von drei Unterseiten (c_0, c_1, c_2), falls die Seite kein Blatt ist, sonst sind alle negativ.
- den positiven externen Index i_{obj} des generischen Objekts, falls die Seite ein Blatt ist, sonst ist dieser negativ.

Für die Suche ergeben sich folgende Anpassungen:

- Die Eingabe der Seiten besteht aus einem Feld aus Seiten, die einzeln durch einen `int8`-Vektor (s. o.) dargestellt sind.
- OpenCL kennt keine dynamische Speicherallokation. Die Queue wird daher als Feld mit vorgegebener Länge modelliert.
- Die fixe Anzahl der Unterseiten ist 3. Dies ergibt sich aus der begrenzten Vektorlänge. Die Nachfolger werden über ihren Index adressiert.

In Abbildung 6.3 sind drei Punkte mit den ganzzahligen Bounding-Boxen dargestellt (Koordinaten in Tabelle 6.2). Die R-Baum-Belegung als `int8`-Vektoren hierzu folgt in Tabelle 6.3.

Im Folgenden werden die Implementierung von T-Netzen und die Berechnung von T-Spline-Flächen basierend auf der eingebetteten Suche in R-Bäumen vorgestellt.

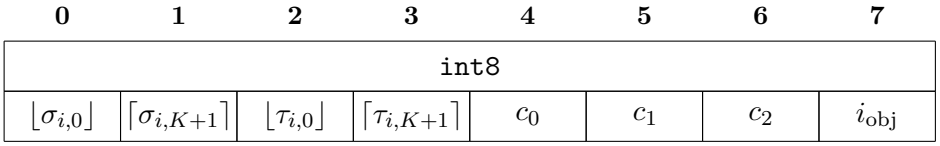
6.2.2 OpenCL-Kodierung von T-Spline-Flächen

Die Erzeugung einer T-Spline-Fläche kann als punktweise Berechnung mit Point-Based-Spline-Flächen (PB-Spline) interpretiert werden. Das T-Netz ist somit für diese Berech-

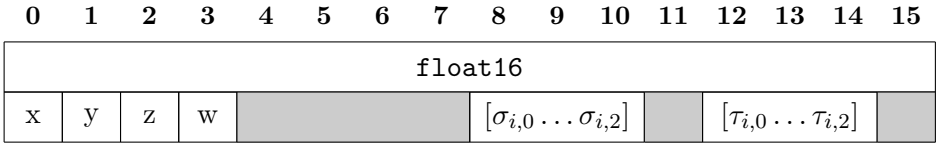
Index	$\lfloor x_{\min} \rfloor$	$\lfloor y_{\min} \rfloor$	$\lceil x_{\max} \rceil$	$\lceil y_{\max} \rceil$	c_0	c_1	c_2	i_{obj}
0	0	0	2	2	1	2	3	-1
1	0	0	1	1	-1	-1	-1	0
2	1	0	2	1	-1	-1	-1	1
3	0	1	1	2	-1	-1	-1	2

Tabelle 6.3: R-Baum-Belegung zum Beispiel

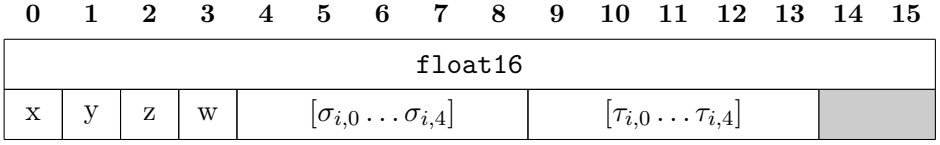
nung irrelevant. Eine PB-Spline-Fläche wird durch eine Menge an gewichteten geometrischen Punkten $v_i := (x, y, z, w)$ und den lokalen Knotenwerten $[\sigma_i, \tau_i]$ beschrieben. Aus Formel (2.8) folgt, dass die Basisfunktionen lediglich im Bereich $\sigma_{i,0} \leq s \leq \sigma_{i,K+1}$ und $\tau_{i,0} \leq t \leq \tau_{i,K+1}$ einen Wert > 0 ergeben. Die Kontrollpunkte werden anhand ihres „Einflussbereichs“, d. h. dem Rechteck $R := [\sigma_{i,0}, \sigma_{i,K+1}, \tau_{i,0}, \tau_{i,K+1}]$, in einem R-Baum abgelegt. Parameterpaare, die außerhalb der lokalen Knotenwerte eines Kontrollpunkts liegen, führen zu Funktionswerten $= 0$ in der T-Spline-Funktion (Formel (2.7)). Diese Kontrollpunkte haben somit keinen Einfluss auf den Flächenverlauf. Ein Algorithmus zur Berechnung von PB-Spline-Flächen ermittelt nun über eine Suche im R-Baum die einflussreichen Kontrollpunkte und berechnet anhand dieser den Flächenpunkt $\mathbf{t}(s, t)$. Das Kodierungsschema einer T-Netz-Zelle im R-Baum folgt der Form



Die referenzierten Kontrollpunkte unterscheiden durch den Grad der T-Spline-Fläche (linear oder kubisch[1]) die Form der separierten Kodierung. Die lineare Belegung ist



und die kubische



Für lineare PB-Spline-Flächen besitzen die lokalen Knotenwerte 2×3 Elemente, für kubische 2×5 Elemente, allgemein $2 \times (K+2)$. Zusätzlich benötigt werden vier Elemente für die Geometrie und das Gewicht des Kontrollpunkts, so dass insgesamt zehn bzw. 14 Elemente in einem `float16`-Vektor belegt sind. Die versetzte Verteilung der Elemente (speziell im linearen Fall) erklärt sich aus dem internen 4-Byte-Alignment[2] der Vektoren.

[1] Höhere Grade sind somit in der umgesetzten Programmierung ausgeschlossen.
[2] Ausrichten der Daten im Speicher auf feste Blockgrößen, um Lese / Schreib-Zugriffe zu optimieren.

Der nachfolgende schematische Algorithmus 6 berechnet für ein Parameterpaar (s, t) den Flächenpunkt $\mathbf{t}(s, t)$ oder NaN[3].

Eingabe: Kontrollpunkte V
begin
– Suche Kontrollpunkte $\hat{V} := \{\hat{v}_i\}$ mit $\sigma_{i,0} \leq s \leq \sigma_{i,K+1}$ und $\tau_{i,0} \leq t \leq \tau_{i,K+1}$
if $\hat{V} \neq \emptyset$ **then**
Ausgabe: $\mathbf{t}(s,t) := \frac{\sum_{\hat{v}_i \in \hat{V}} w_i \cdot \mathbf{c_i} \cdot N^K[\sigma_i,\tau_i](s,t)}{\sum_{\hat{v}_i \in \hat{V}} w_i \cdot N^K[\sigma_i,\tau_i](s,t)}$
else
Ausgabe: NaN
end
end

Algorithmus 6 : Berechnung von PB-Spline-Flächen

Der Algorithmus berechnet somit die PB- bzw. T-Spline-Fläche aus einer reduzierten Menge der Kontrollpunkte. Eine OpenCL-Implementierung verwendet als Eingabe den gegebenen Parameterpunkt (`float2`), das T-Netz als `int8`-Feld mitsamt dem Index der Wurzel und die `float16`-Vektoren für die Kontrollpunkte. Dabei entsteht als Ausgabe ein `float4`-Vektor mit homogenen Geometriekoordinaten.

Die Suche der Kontrollpunkte ist eine wichtige und zeitkritische Aufgabe. Wie vorab beschrieben ist sie hier als Subroutine eingebettet. Die PB-Spline-Funktion summiert die gefundenen Kontrollpunkte oder erzeugt eine ungültige Ausgabe. Die Fallunterscheidung kann entfallen, sofern eine Null-Division als *NaN* definiert ist[4].

Der vorgestellte Algorithmus beruht auf einer Suche mit logarithmischer Tiefe und bietet nur bei einer hohen Anzahl an Kontrollpunkten einen spürbaren Vorteil. Andernfalls werden sämtliche Kontrollpunkte in direkter Abfolge summiert.

Im folgenden Abschnitt werden die OpenCL-Programme in den Prozess der Host-Device-Kommunikation integriert.

6.3 Parallele Algorithmen mit OpenCL

Die eingesetzten Datenstrukturen und darauf definierte Algorithmen werden nun in effiziente OpenCL-Implementierungen umgewandelt. Daher werden die kritische Teilschritte dieser Arbeit auf ihre Parallelität hin untersucht und auf das OpenCL-Programmiermodell abgebildet.

Allgemein berücksichtigt die gesamte Prozesskette der Bathymetrie-Reduktion stets Konzepte der Parallelprogrammierung. Beispiele hierfür sind einfache Map-Reduce-Operationen wie die Berechnung von Minima / Maxima oder die effektive Sortierung in Datenstrukturen.

6.3.1 Berechnung von Punkten auf T-Spline-Flächen

Die Berechnung eines Punkts auf einer T-Spline-Fläche ist eine in diesem Kontext häufig wiederkehrende Operation. Sie ist beispielsweise Bestandteil in der Berechnung

[3] Not A Number – Ein Format für eine ungültige Gleitkommazahl
[4] Auf diversen OpenCL-unterstützenden Plattformen ist dies als Compiler-Option konfigurierbar.

von Differenzfehlern oder der Visualisierung. Hier erfolgt das Berechnen von Flächenpunkten entlang eines hochaufgelösten, über die Parameterrichtungen aufgespannten Gitters. Das Ergebnis ist ein dichtes Raster aus Flächenpunkten im dreidimensionalen Raum, das optional durch bilineare Interpolation weiter verfeinert wird. Die Berechnung der einzelnen Punkte erfolgt parallel. Die Parallelität ergibt sich aus der Anzahl der zu berechnenden Flächenpunkte (üblich sind 1 – 10 Mio. Punkte).

Krishnamurthy et al. stellen Lösungsansätze für die Berechnung von NURBS-Flächen mit *Shader*[5]-Programmen und GPGPU-Bibliotheken vor [47] [48]. Weitere Anwendungen für geometrische Operationen mit GPU-Unterstützung finden sich in [49] [50].

Eine T-Spline-Fläche wird von ihrem zugrundeliegenden T-Netz losgelöst und als PB-Spline-Fläche interpretiert. Der schematische Algorithmus für einen einzelnen Flächenpunkt ist in Abschnitt 6.2.2 beschrieben. Dieser wird direkt parallelisiert, da die einzelnen Flächenpunkte unabhängig voneinander sind. Abbildung 6.4 zeigt die Integration der OpenCL-Schnittstelle in die T-Spline-Flächenberechnung. Der vorgestellte Algorithmus für die Generierung von PB-Spline-Flächen geht als Kernel-Programm in den gesamten Prozess ein.
Der Lebenslauf eines OpenCL-Programms besteht aus den folgenden Schritten:

- Die **Initialisierung** des OpenCL-Context umfasst zu Beginn die Suche, Auswahl und Initialisierung des verfügbaren, OpenCL-unterstützenden Device.
- Die **Kompilierung** erzeugt ein OpenCL-Programm zur Laufzeit (oder liest dieses als Binärcode ein).
- Der OpenCL-Context **erzeugt einen Kernel** und stellt diesen auf dem Device bereit.
- Das **Anlegen der Buffer** für Host- und Device-Speicher. Diese sind erforderlich für den Datentransfer der Eingabe und des Ergebnisses. Buffer können als Input und / oder Output-Speicher angelegt werden. Hierfür erzeugt und belegt der Host die benötigten Datenfelder für die T-Netze und die Kontrollpunkte.
- Das **Setzen der Kernel-Argumente** und der Transfer zum Kernel. Für die PB-Spline-Flächen sind dies:
  - die Seiten (`nodes`) im R-Baum für die T-Netz-Zellen als `int8[]`.
  - die Kontrollpunkte (`vertices`) als `int16[]`.
  - die parametrische rechteckige Dimension des T-Netzes für die Umrechnung des lokalen Index eines Work Items in ein Parameterpaar.
  - die Größe der Ausgabe, d. h. die Dimension des Rasters für die Flächeninterpolation.
  - der Ausgabespeicher der homogenen Flächenpunkte als `int4[]`.
- Der **Aufruf des Kernels** übergibt den Kontrollfluss zunächst an das OpenCL-Device. Jedes Work Item berechnet anhand seiner Indexposition einen Parameterpunkt und schreibt sein Ergebnis in die Ausgabe.
- Die **Finalisierung** des Programms. Zum Schluss wartet der Host auf das Beenden der Device-Operationen und liest das Ergebnis aus.

[5] GPU-Programme für Pixel-Operationen

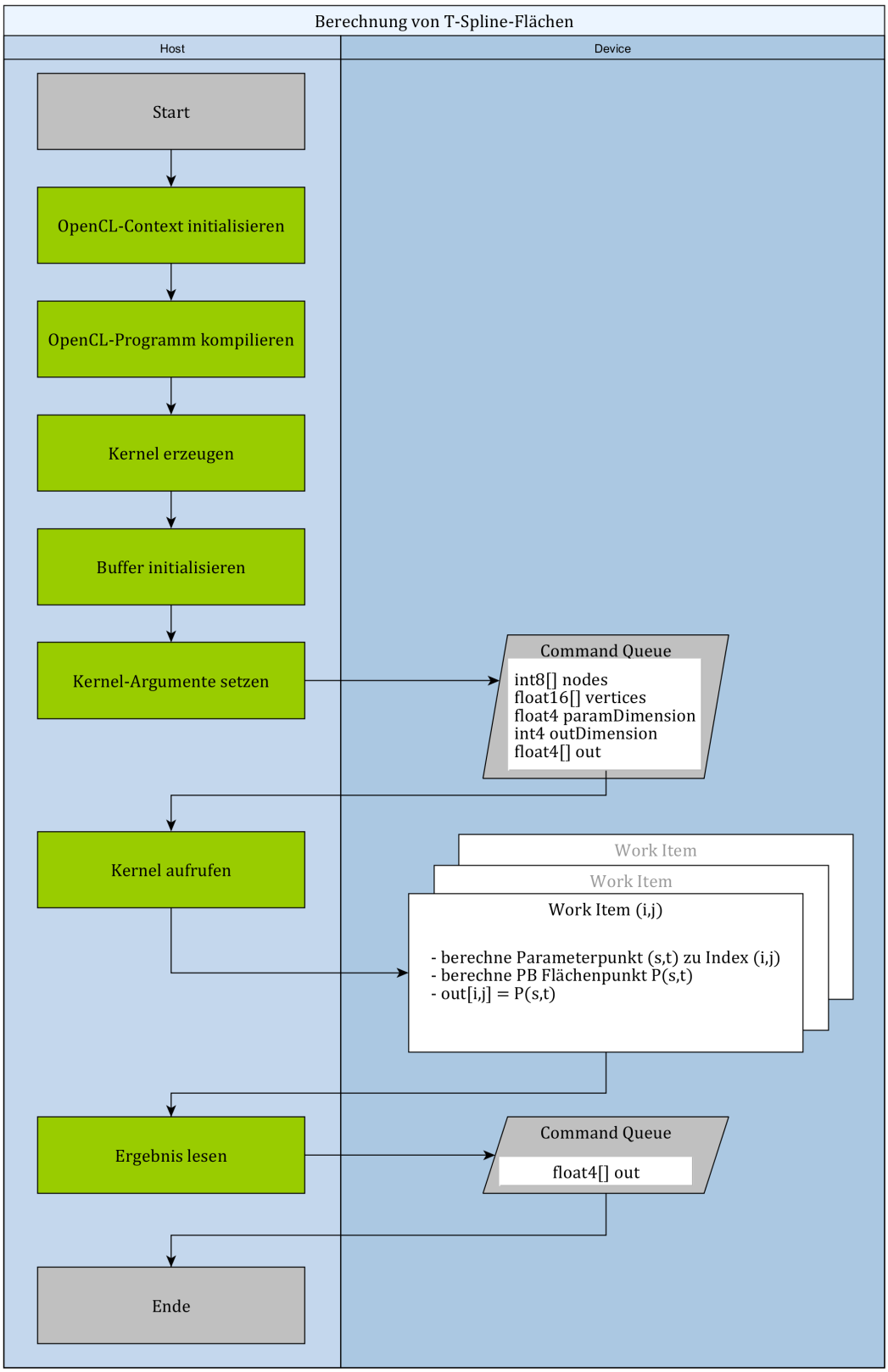


Abbildung 6.4: T-Spline-Flächenberechnung mit OpenCL

Losgelöst von der Host-Device-Interaktivität können die benötigten Datenfelder, die als Kernel-Argumente übergeben werden, bereits vorab erzeugt werden. Dies umfasst die Host-seitige Erzeugung und Initialisierung von ausreichend großen Speicherblöcken für die Felder der T-Netze und der Kontrollpunkte. In den folgenden Abschnitten wird auf die weitere Beschreibung des vollständigen Prozesses verzichtet und somit der Fokus auf den zentralen Datentransfer und die Kernel-Methoden gelegt.

6.3.2 Knotenherleitung in T-Netzen

Die Knotenherleitung im T-Netz wird auf den linearen, horizontalen Fall (d. h. in s-Parameterrichtung) vereinfacht und die damit verbundene OpenCL-Integration erläutert.

Horizontale Knotenherleitung für lineare T-Spline-Flächen

Algorithmus 7 berechnet die Knotenwerte für einen Kontrollpunkt aus einem gegebenen T-Netz. Die Beschreibung erfolgt zunächst nur für den horizontalen Fall bei linearem Ansatzgrad entlang der s-Parameterrichtung. Die vertikalen Knotenwerte ergeben sich mit den entsprechenden Anpassungen. Die wesentlichen Änderungen für höhere Ansatzgrade folgen im Anschluss.

Eingabe:
– Parameterpaar (s_i, t_i) des Kontrollpunkts v_i
– Ein T-Netz für eine lineare T-Spline-Fläche
begin
  – Initialisiere $\sigma_i := [?, s_i, ?]$ (? = unbekannter Wert)
  – Bestimme die horizontale Schnittlinie $l := (*, t_i)$
  – Suche die nächste geschnittene vertikale Kante $e := (s_W, *)$ in West-Richtung
  – setze $\sigma_{i,0} := s_W$
  – Suche die nächste geschnittene vertikale Kante $e := (s_O, *)$ in Ost-Richtung
  – setze $\sigma_{i,2} := s_O$
  if $\sigma_{i,0}$ *unbekannt* **then**
    – setze $\sigma_{i,0} := \sigma_{i,1}$
  end
  if $\sigma_{i,2}$ *unbekannt* **then**
    – setze $\sigma_{i,2} := \sigma_{i,1}$
  end
end
Ausgabe: $\sigma_i = [\sigma_{i,0}, \sigma_{i,1}, \sigma_{i,2}]$

Algorithmus 7 : Herleitung von linearen, horizontalen Knotenwerten im T-Netz

Die Eingabe besteht lediglich aus dem Parameterpaar des aktuellen Kontrollpunkts und dem vollständigen T-Netz.

Zu Beginn wird die mittlere Position der Knotenwerte mit dem horizontalen Parameterwert des Kontrollpunkts initialisiert. Im Anschluss erfolgt die Schnittberechnung entlang einer horizontalen Linie in westlicher bzw. östlicher Richtung ausgehend vom Kontrollpunkt. Bei horizontalen Knotenwerten werden ausschließlich vertikale Kanten gesucht und das Ergebnis auf die jeweils nächste Kante reduziert. Wird letztlich eine Kante in West-Richtung gefunden, so wird ihr s-Wert als erster Knotenwert $\sigma_{i,0}$ übernommen. Analog ergibt der Schnitt in östlicher Richtung den letzten Knotenwert $\sigma_{i,2}$.

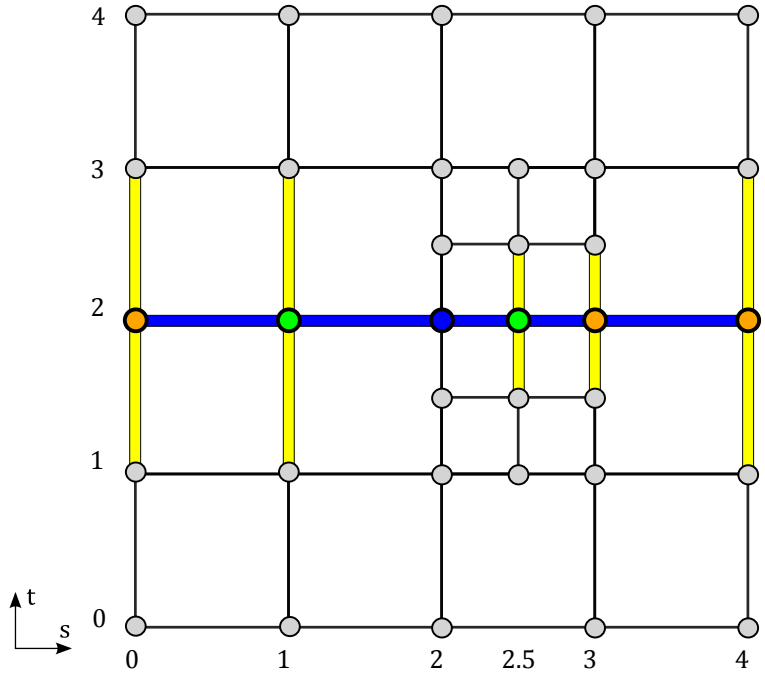


Abbildung 6.5: Schnittberechnung im T-Netz: dargestellt sind der Kontrollpunkt (blau), die geschnittenen vertikalen Kanten (gelb) mit ihren Schnittpunkten (orange) und die explizit nächsten Schnittpunkte (grün). Somit lauten die horizontalen Knotenwerte für lineare T-Spline-Flächen $(1, 2, 2\frac{1}{2})$.

Die beiden verbleibenden Schritte prüfen, ob die Knotenwerte tatsächlich gefunden werden, und fügen ggf. den mittleren Wert über Knotenwertverdopplung hinzu. Dies kann erforderlich sein, da am Rand des T-Netzes kein Schnitt in der jeweiligen Richtung gefunden wird. Abbildung 6.5 fasst dies an einem einfachen Beispiel zusammen.

Es ist offensichtlich, dass die lokalen Knotenwerte isoliert und somit parallel zu jedem einzelnen Kontrollpunkt im T-Netz berechnet werden. Daher ergibt sich die Parallelität direkt aus der Anzahl der Kontrollpunkte im T-Netz (im Beispiel aus Kapitel 7 über 60.000 Kontrollpunkte).

Knotenherleitung für kubische und höhergradige T-Spline-Flächen

Strukturell wird der im vorherigen Abschnitt beschriebene Algorithmus übernommen. Anpassungen ergeben sich daraus, dass für lokale Knotenwerte bei Grad K genau $k := (K + 1)/2$ Schnitte in jede Richtung berechnet werden. Der Algorithmus wird um eine Datenstruktur mit einer maximalen Länge von k ergänzt, in denen die ermittelten Schnittkoordinaten jeder Richtung aufsteigend (für östliche und nördliche Richtung) bzw. absteigend (westlich und südlich) einsortiert werden. Liegt eine neu gefundene Schnittkoordinate außerhalb des bereits gefundenen Intervalls, so verfällt diese, andernfalls wird sie in die Datenstruktur einsortiert. Besitzt die Datenstruktur bereits die maximale Größe, so wird der letzte (d. h. in der Ordnung größte oder kleinste) Knotenwert gelöscht. Besitzt die Datenstruktur nach Abschluss der Suche nicht die benötigte Anzahl an Knotenwerten, dann werden Knotenwerte verdoppelt, bis die Länge k erreicht ist.

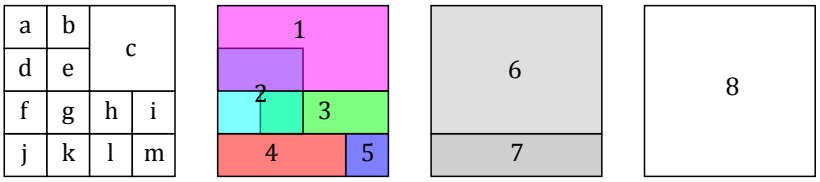


Abbildung 6.6: Zerlegung eines T-Netzes in seine Bounding-Boxen

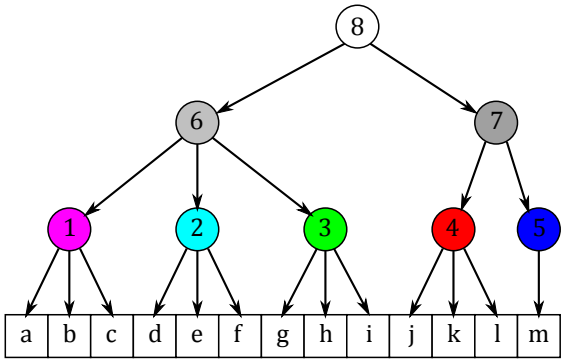


Abbildung 6.7: R-Baum-Struktur zu Abbildung 6.6

Datenstruktur für Zellen im T-Netz

Die Berechnung der Knotenwerte erfolgt direkt auf einem T-Netz. Hierfür werden die Zellen des T-Netzes innerhalb eines generischen R-Baums eingebettet. Anfragen an den R-Baum, welche Zellen ein Schnittstrahl trifft, werden somit effizient durchgeführt.

Das beschriebene Kodierungsschema für generische R-Bäume wird direkt übernommen (Abschnitt 6.2.1). Die externen Objekte bestehen in diesem Fall aus den Zellen im T-Netz und ihren reellen Koordinaten. Diese lassen sich einzeln als `float4`-Vektor abspeichern:

0	**1**	**2**	**3**
`float4`			
$x_{\min}$	$y_{\min}$	$x_{\max}$	$y_{\max}$

Die Bounding-Boxen in den R-Baum-Seiten entsprechen den ganzzahlig gerundeten Eckpunktkoordinaten der T-Netz-Zellen. Die Zerlegung eines T-Netzes in Bereiche mit drei Nachfolgern und der sich daraus ergebende R-Baum sind in den Abbildungen 6.6 bzw. 6.7 beispielhaft dargestellt.

Abbildung 6.8 verkürzt den schematischen Ablauf der OpenCL-basierten Knotenherleitung auf die wesentlichen Operationen der Command Queue. Die folgenden Kernel-Argumente werden gesetzt:

- ein `int8[]`-Feld für die Kodierung der R-Baum-Seiten (`nodes`)
- ein `float4[]`-Feld für die T-Netz-Zellen (`cells`)
- ein `float2[]`-Feld mit den Parameterwerten der Kontrollpunkte (`paramPoints`)
- ein `float4[]`-Feld für die Ausgabe der linearen bzw. `float8[]` für die kubischen Knotenwerte (s. u.).

Das lineare Ergebnis für einen Kontrollpunkt mit dem Parameterpaar (s, t) ist das 4er-Tupel $(\sigma_{i,0}, \sigma_{i,2}, \tau_{i,0}, \tau_{i,2})$. Nach dem Ergänzen der fehlenden mittleren Werte ergeben sich die Knotenwerte $\sigma_i := (\sigma_{i,0}, s, \sigma_{i,2})$ und $\tau_i := (\tau_{i,0}, t, \tau_{i,2})$.

Die kubische Lösung erzeugt das 8er-Tupel $(\sigma_{i,0}, \sigma_{i,1}, \sigma_{i,3}, \sigma_{i,4}, \tau_{i,0}, \tau_{i,1}, \tau_{i,3}, \tau_{i,4})$ und folglich $\sigma_i := (\sigma_{i,0}, \sigma_{i,1}, s, \sigma_{i,3}, \sigma_{i,4})$ und $\tau_i := (\tau_{i,0}, \tau_{i,1}, t, \tau_{i,3}, \tau_{i,4})$.

6.3.3 Dragging

Das Dragging-Verfahren aus Abschnitt 4.6 ermöglicht die Flächenrückführung durch die progressive Verschiebung einzelner Kontrollpunkte. D. h. es werden die einzelnen Schritte bis zum Erreichen eines globalen Abbruchkriteriums wiederholt. Üblich sind das Erreichen einer vorgegebenen Anzahl an Iterationen oder das Unterschreiten eines maximal geduldeten Fehlers.

Im Kern erfolgt die Optimierung der Kontrollpunkte der $(k + 1)$-ten Stufe durch die Berechnung der Menge der Dragging-Vektoren Δ^k, und es folgt $C^{k+1} := C^k + \Delta^k$. Dies erlaubt eine parallele Berechnung der Verschiebung für jeden Kontrollpunkt. Allerdings muss eine explizite Synchronisation der einzelnen Schritte stattfinden: nach jeder Iteration werden die neuen Dragging-Vektoren auf die bisherigen Geometriekoordinaten der Kontrollpunkte addiert und anschließend der maximale Fehler zwischen der modifizierten T-Spline-Fläche und der Bathymetrie bestimmt.

Die Anzahl der Kontrollpunkte gibt die Parallelität vor. Diese kann (wie im Zusammenhang mit der Knotenwertherleitung erwähnt) einige Zehntausend Kontrollpunkte erreichen.

Implementierung

Der Ablauf ist in Abbildung 6.9 dargestellt. Die Eingabe besteht aus :

- den R-Baum-Seiten des T-Netzes der aktuellen Iteration (`nodes`).
- den Kontrollpunkten des T-Netzes (`vertices`).
- einer Diskretisierung der Referenzfläche als OpenCL-`image2d` (`refSurface`).

Die Berechnung von Flächenpunkten aus dem T-Netz verwendet einen ähnlichen Ansatz wie die Flächeninterpolation mit PB-Spline-Flächen. Die Referenzfläche wird als diskretisiertes Raster mit Tiefenwerten vorberechnet, da sie sich im Ablauf des Algorithmus nicht ändert. Fehlende Zwischenwerte können aus den `image2d`-Daten sehr effektiv über *Sampler* bilinear interpoliert werden. Die Ausgabe liefert letztlich die gesamten Dragging-Vektoren Δ^{k+1} der $(k + 1)$-ten Iteration.

Host-Synchronisation

Das OpenCL-Programmiermodell sieht in der Version 2.1 keine globale Datensynchronisation durch Barrieren vor[6] (Abschnitt 6.1.4). Verschiebungsvektoren des Dragging-Verfahrens werden Device-intern nur auf den Ausschnitt der Geometriekoordinaten der Kontrollpunkte addiert, die in der gleichen Work Group bearbeitet werden. Daher muss die Synchronisation extern im CPU-Kontext des Host erfolgen. Aus diesem Grund werden die Verschiebungsvektoren Δ^{k+1} nach jeder Iteration aus dem Device- in den

[6] Stand von September 2015

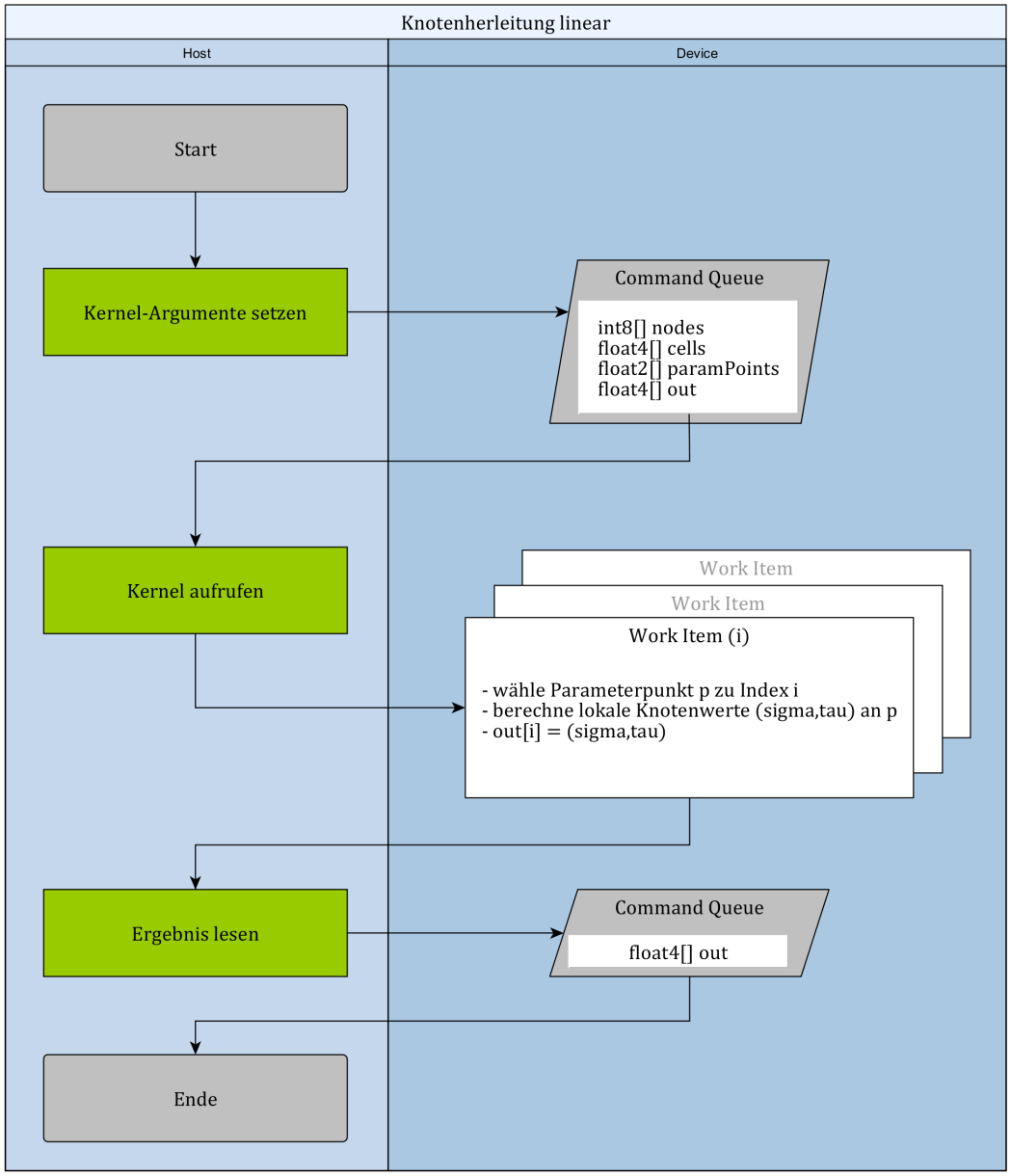


Abbildung 6.8: Knotenherleitung mit OpenCL

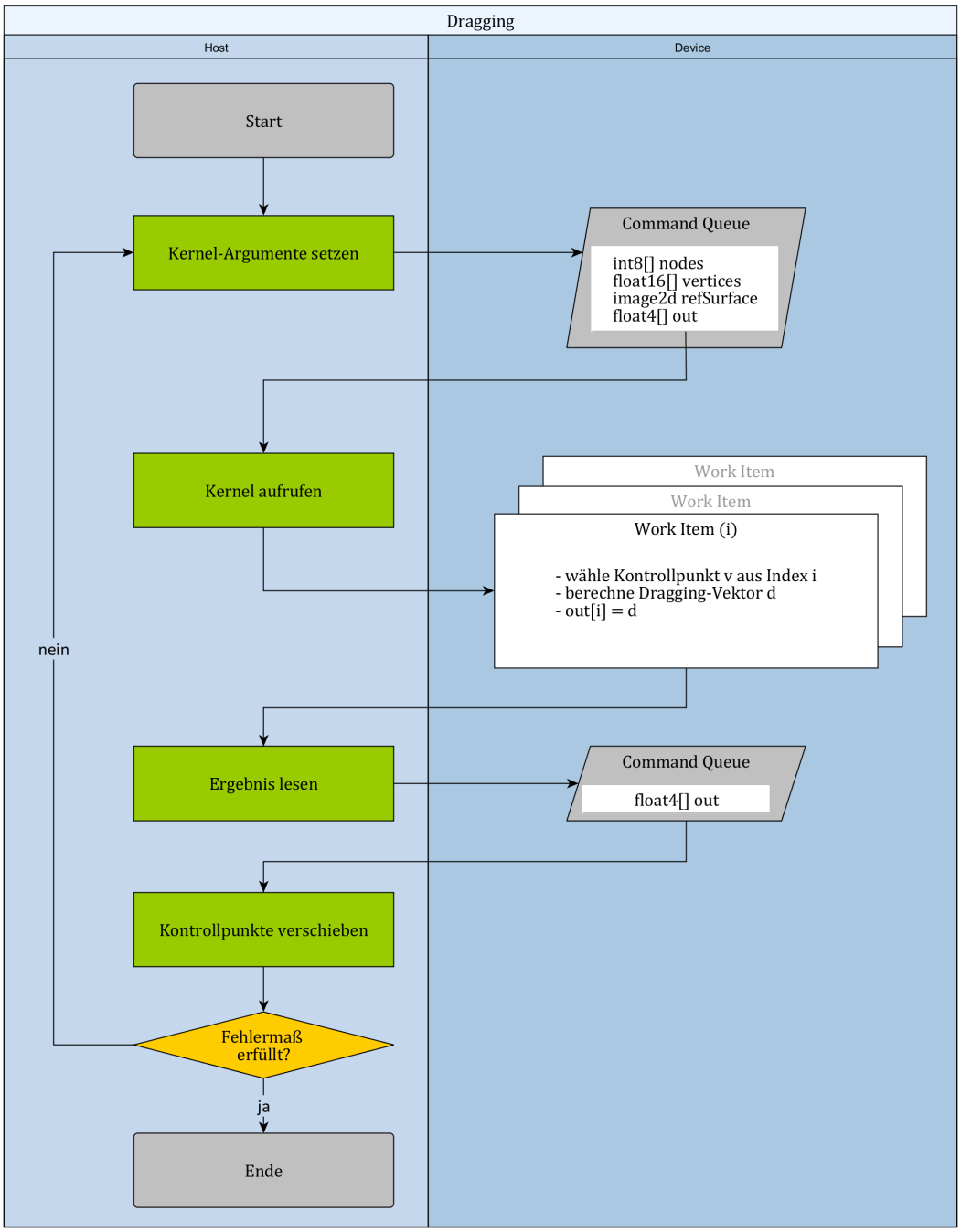


Abbildung 6.9: Dragging mit OpenCL

Host-Speicher kopiert und die Geometrie der Kontrollpunkte wird um die Dragging-Vektoren verschoben. Anschließend schreibt der Host die veränderten Kontrollpunkte zurück in den Device-Speicher und der Dragging-Kernel wird erneut aufgerufen.

6.3.4 Lineare Optimierung

Die Optimierung der Geometrie von Kontrollpunkten erfolgt über die Methode der kleinsten Quadrate. In Abschnitt 4.5.2 wird die Modellmatrix T definiert. Die Berechnung ihrer Matrixeinträge ist elementweise möglich und somit hochgradig parallelisierbar. Eine wesentliche Beschleunigung ergibt sich aus der simultanen Berechnung der Werte der T-Spline-Basisfunktion für jeden Kontrollpunkt. Die Implementierung orientiert sich stark an den vorherigen Programmen, so dass an dieser Stelle auf eine detaillierte Beschreibung der OpenCL-Integration verzichtet wird (Abbildung 6.10). Weitere Besonderheiten bzgl. Synchronisation oder der Einbettung von Subroutinen entfallen in dieser Anwendung.

6.4 Zwischenstand

Großräumige Bathymetrien mit dichten Datenmengen sind mit den vorgestellten, allerdings unbeschleunigten Verfahren nur bei geringen Größen handhabbar (max. einige Tausend Kontrollpunkte). Unter Zuhilfenahme der OpenCL-Schnittstelle wird eine hochperformante Implementierung für die Reduktion bathymetrischer Flächen erwirkt. Im letzten Teil der Arbeit folgt die Betrachtung einer realen Bathymetrie, deren Validierung und Praxistauglichkeit erst durch die eingearbeitete GPGPU-Beschleunigung ermöglicht wird.

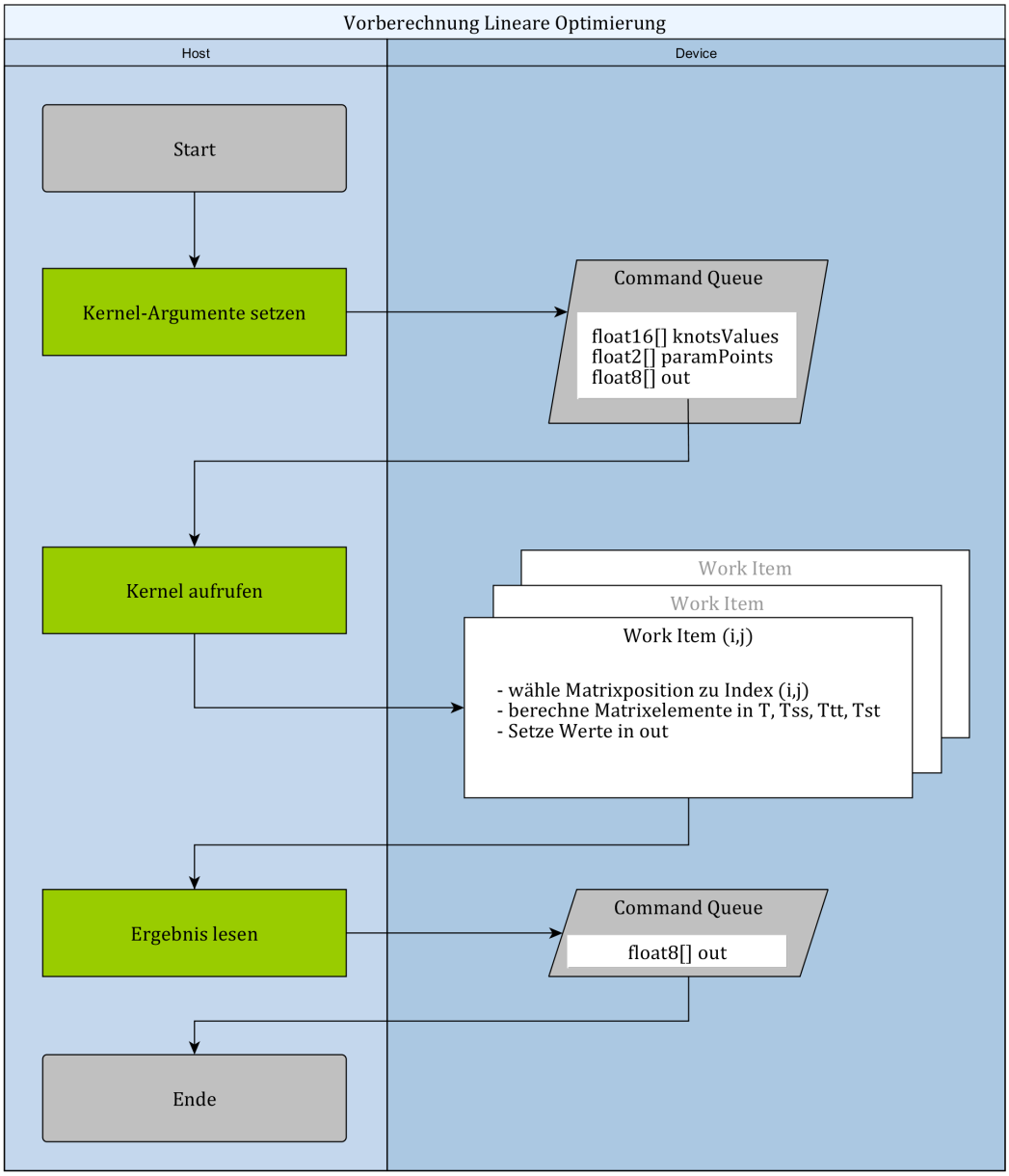


Abbildung 6.10: Vorberechnung der linearen Optimierung mit OpenCL

7 Analyse

Eine Validierung bewertet die Reduktion bathymetrischer Flächen anhand einer realen Bathymetrie. Der Fokus liegt hierbei auf der Skalierung der Daten und der Durchführbarkeit der Methoden unter Berücksichtigung des offensichtlichen Vorteils der OpenCL-Unterstützung. Die Reduktion der Bathymetrie greift auf die vorgestellten Optimierungsverfahren (lineare Optimierung und Dragging) zurück. Diese erzielen praxisnahe Ergebnisse. Weitere Berücksichtigung finden zusätzliche Vergleiche der Netzgröße, der Approximationsgüte und der Fehlerklassenverteilung. Die Darstellung der erzeugten T-Spline-Flächen ermöglicht die visuelle Begutachtung.

7.1 Reale Bathymetrie

Die Abbildung 7.1 zeigt eine Bathymetrie aus dem Jahr 2009 für die Fahrrinne der Weser bei Bremerhaven. Die Bathymetrie beschreibt eine Kartierung der Fahrrinne und deckt weitere umliegende Flachwassergebiete ab. Die Messgebiete entstammen verschiedenen Sonarpeilungen, und die landnahen Gebiete beruhen auf Laserscans. Der Testdatensatz umfasst ein Gebiet von ca. 51,7 km × 41,0 km (entspricht 2119 km^2) mit Rasterpunkten im Abstand von 1 m. Die Tiefenwerte liegen in einem Intervall zwischen ca. -25 m bis +5 m.

In Anlehnung an das vereinfachte Beispiel aus Abschnitt 3.3 wird ein vollständiger Durchlauf der indirekten Reduktion vorgeführt. Hierfür werden sowohl die lineare Optimierung mit der Methode der kleinsten Quadrate als auch das Dragging-Verfahren untersucht. Ein Vergleich stellt die beiden Durchläufe hinsichtlich visueller und objektiver Kriterien nebeneinander dar.

7.1.1 Indirekte Reduktion

Aufgrund der gedrehten Lage und der ungleichmäßig abgedeckten Vermessung der Fahrrinne und ihrer Ausläufer stellt die indirekte Reduktion die bessere Wahl dar, da sie die vorherige Ausrichtung der Bathymetrie ermöglicht. In Abbildung 7.2 ist die transformierte Bathymetrie nach dem Warping-Schritt dargestellt. Das Ergebnis der zusätzlichen Dilatation zeigt Abbildung 7.3. Es ergibt sich ein neues Gebiet von etwa 65,7 km × 13,2 km. Dies entspricht einer Fläche von 867 km^2, somit etwa 40% der ursprünglichen Bathymetrie.

7.1.2 Lineare Optimierung

Der stufenweise Aufbau der T-Netze für die Schritte 0 bis 5 für lineare und kubische T-Spline-Flächen ist dargestellt in den Abbildungen 7.10 f. und 7.12 f. in Abschnitt 7.3. Die Kenndaten der stufenweisen Netzverfeinerung mit linearen und kubischen T-Spline-Flächen stellt Tabelle 7.1 gegenüber (siehe auch Abbildung 7.5). Diese beinhalten die Anzahl der Kontrollpunkte als Größe des T-Netzes, die mittlere Abweichung, die dazugehörige positive Standardabweichung und den Median der Differenzfehler. Weitere statistische Angaben können den Histogrammen der Tiefenwert-Fehlerklassen entnommen werden (Abbildung 7.4).

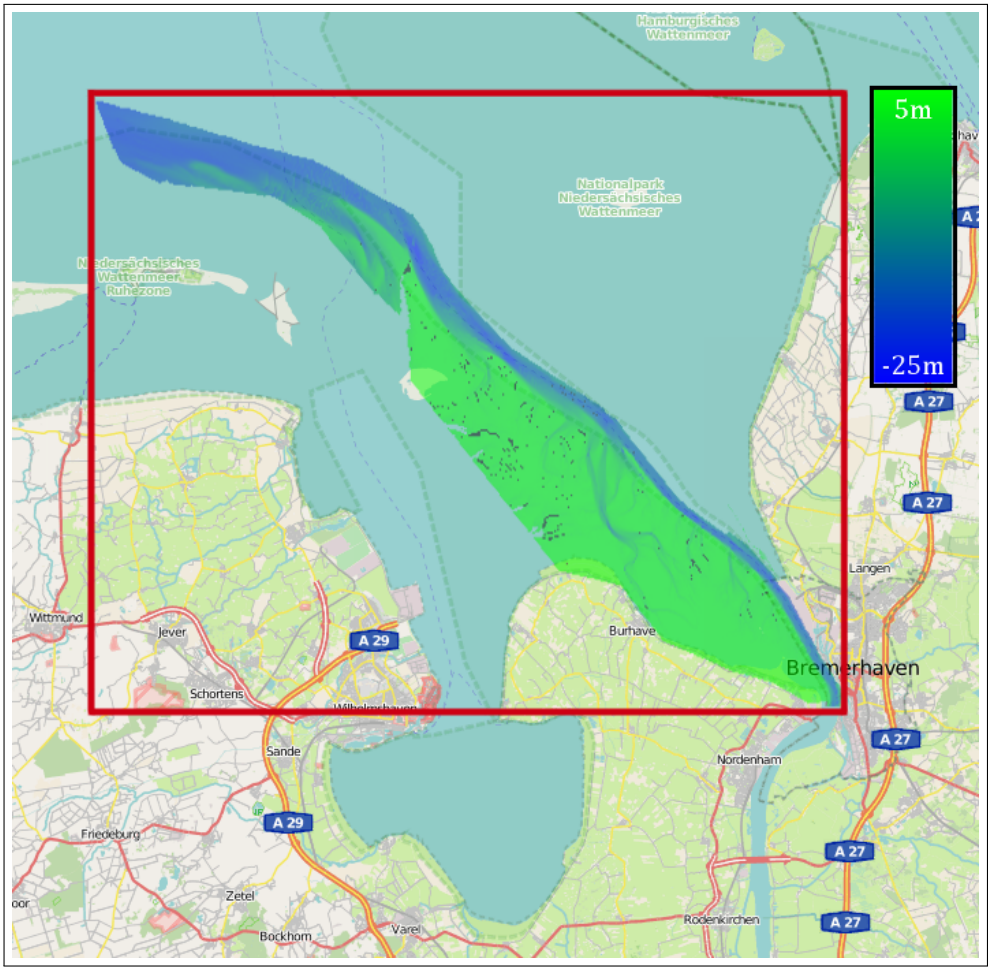


Abbildung 7.1: Räumliche Einordnung der Bathymetrie der Weser-Fahrrinne bei Bremerhaven. Das hervorgehobene Gebiet hat eine Größe von ca. 2119 km^2. (Die unterliegende Karte stammt aus der OpenStreetMap unter der CC-BY-SA 2.0-Lizenz.)

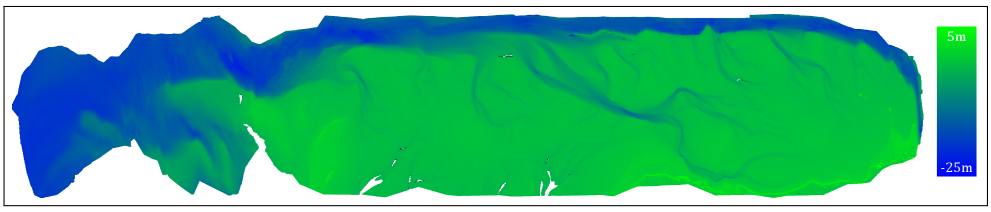


Abbildung 7.2: Transformierte Bathymetrie nach Warping

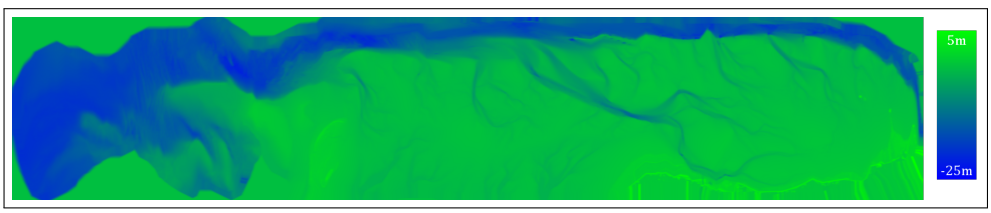


Abbildung 7.3: Transformierte Bathymetrie nach Warping und Dilatation

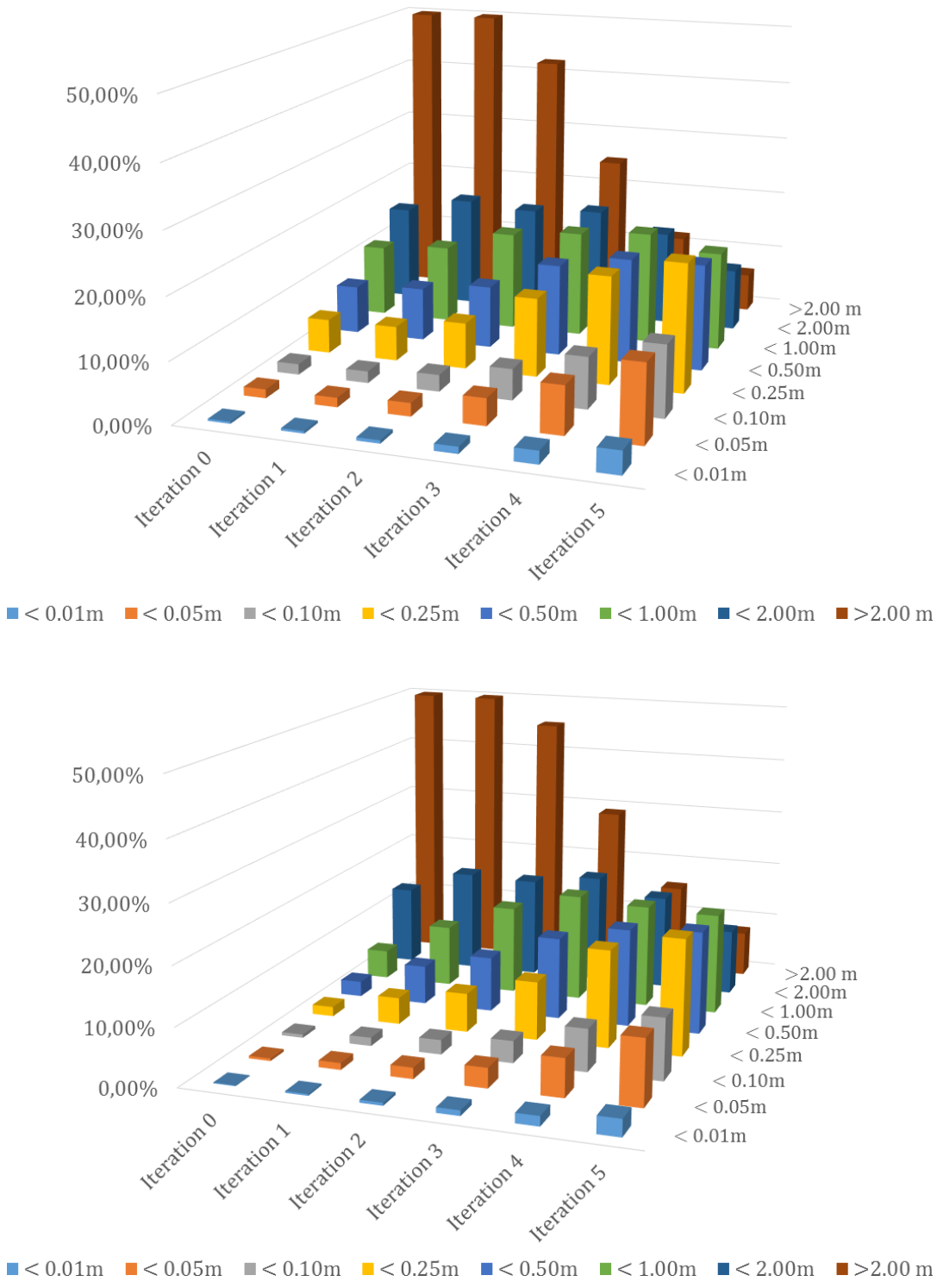


Abbildung 7.4: Histogramme der Differenzfehler linearen Optimierung für lineare (oben) und kubische (unten) T-Spline-Flächen

Iteration	Grad	Kontroll-punkte	Mittelwert [m]	Standard-abweichung [m]	Median [m]
0	1	64	1,74	0,91	1,59
	3	64	2,14	0,67	1,83
1	1	256	1,67	0,91	1,50
	3	112	1,80	0,87	1,61
2	1	688	1,50	0,94	1,17
	3	412	1,56	0,93	1,30
3	1	1888	1,11	0,92	0,61
	3	1420	1,23	0,92	0,71
4	1	5776	0,77	0,79	0,33
	3	4924	0,87	0,84	0,38
5	1	19528	0,55	0,66	0,18
	3	17932	0,61	0,71	0,23

Tabelle 7.1: Approximationsgüte bei linearer Optimierung

Die lineare Optimierung (basierend auf der Methode der kleinsten Quadrate) korrigiert in jeder Iteration die Geometrie der Kontrollpunkte. Die daraus entstehenden Fehler weisen die Qualität der Approximation aus. Das Approximationsschema durchläuft, ausgehend vom initialen T-Netz 0-ter Stufe, insgesamt fünf Iterationen. Es bricht allerdings nach weiteren Stufen aufgrund der hohen Dimension der Optimierungsmatrix und des damit verbundenen Speicherplatzbedarfs ab. Die entstehenden Matrizen sind mit der verwendeten Numerik-Softwarebibliothek nicht mehr effektiv zu handhaben und die berechneten Lösungen weisen Instabilitäten auf.

Nach Abschluss der Optimierung steht ein T-Netz für eine lineare bzw. kubische T-Spline-Fläche zur Verfügung. Das allgemeine Verfahren der indirekten Reduktion erzeugt zunächst eine T-Spline-Fläche mit einer größeren Ausdehnung als die Bathymetrie. Zur Korrektur wird eine Maskierung der erzeugten Fläche durch einen logischen Schnitt mit der ursprünglichen Bathymetrie durchgeführt, um die relevanten Gebiete auszustanzen (Abbildung 7.6, dazugehörige maskierte Darstellung in Abbildung 7.18).

Die Abbildungen 7.14 bis 7.16 zeigen die erzeugten linearen T-Spline-Flächen für die Stufen drei bis fünf, Abbildung 7.17 bis 7.19 jeweils für kubische T-Spline-Flächen. Dargestellt sind die Flächen und die Klassifikation der Differenzfehler im Vergleich mit der Originalbathymetrie.

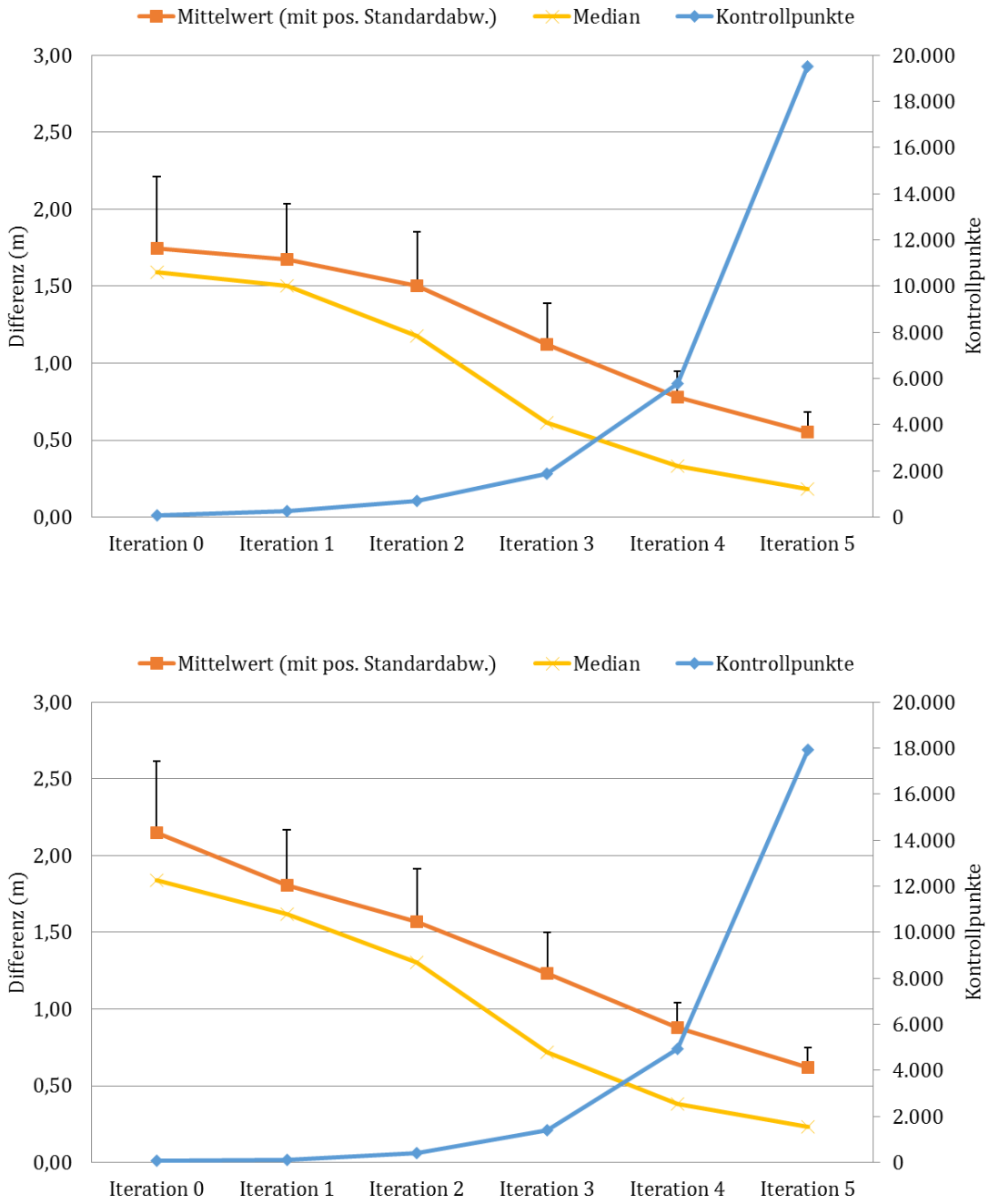


Abbildung 7.5: Vergleich der Qualität der linearen Optimierung in der Bathymetrie (linear und kubisch)

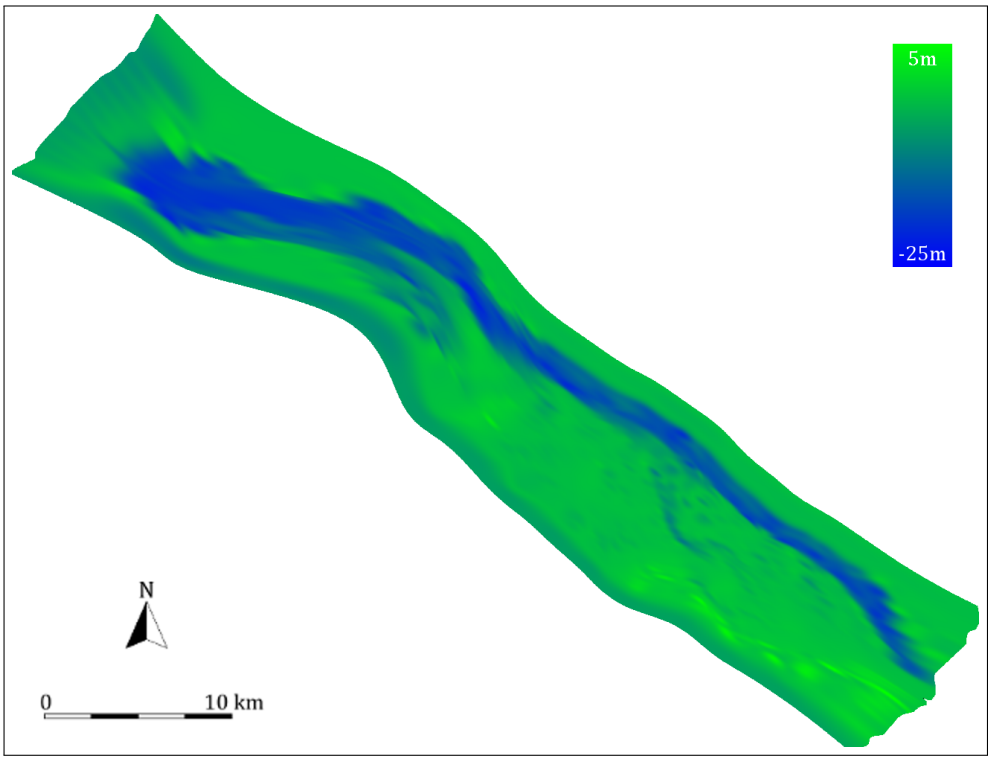


Abbildung 7.6: T-Spline-Fläche ohne Maskierung mit der Originalbathymetrie

Iteration	Grad	Kontroll-punkte	Mittelwert [m]	Standard-abweichung [m]	Median [m]
0	1	64	1,76	0,90	1,60
	3	64	2,18	0,64	1,86
1	1	208	1,68	0,91	1,51
	3	112	1,80	0,87	1,61
2	1	556	1,50	0,95	1,17
	3	412	1,56	0,93	1,30
3	1	1684	1,13	0,93	0,62
	3	1420	1,23	0,92	0,71
4	1	5284	0,80	0,81	0,33
	3	4924	0,87	0,84	0,38
5	1	18064	0,56	0,68	0,18
	3	18012	0,61	0,71	0,23
6	1	66620	0,42	0,60	0,12
	3	63320	0,45	0,61	0,13

Tabelle 7.2: Approximationsgüte des Dragging-Verfahrens

7.1.3 Dragging-Verfahren

Tabelle 7.2 stellt die Netzgrößen und Qualitätsmerkmale der einzelnen Approximationsschritte für das Dragging-Verfahren dar (siehe auch Abbildung 7.7). Das Approximationsschema verläuft sichtbar stabiler, und letztlich entstehen weitaus feinere T-Netze.

Der Entstehungsprozess der T-Netze ist dargestellt in den Abbildungen 7.20 ff. und 7.23 ff. Fortgeschrittene Iterationen des Dragging-Verfahrens sind in den Abbildungen 7.26 bis 7.29 dokumentiert. In den Abbildungen 7.30 bis 7.33 ist das Dragging-Verfahren mit kubischen Basisfunktionen dargestellt. Die Histogramme der Fehlerklassen sind ersichtlich in Abbildung 7.8.

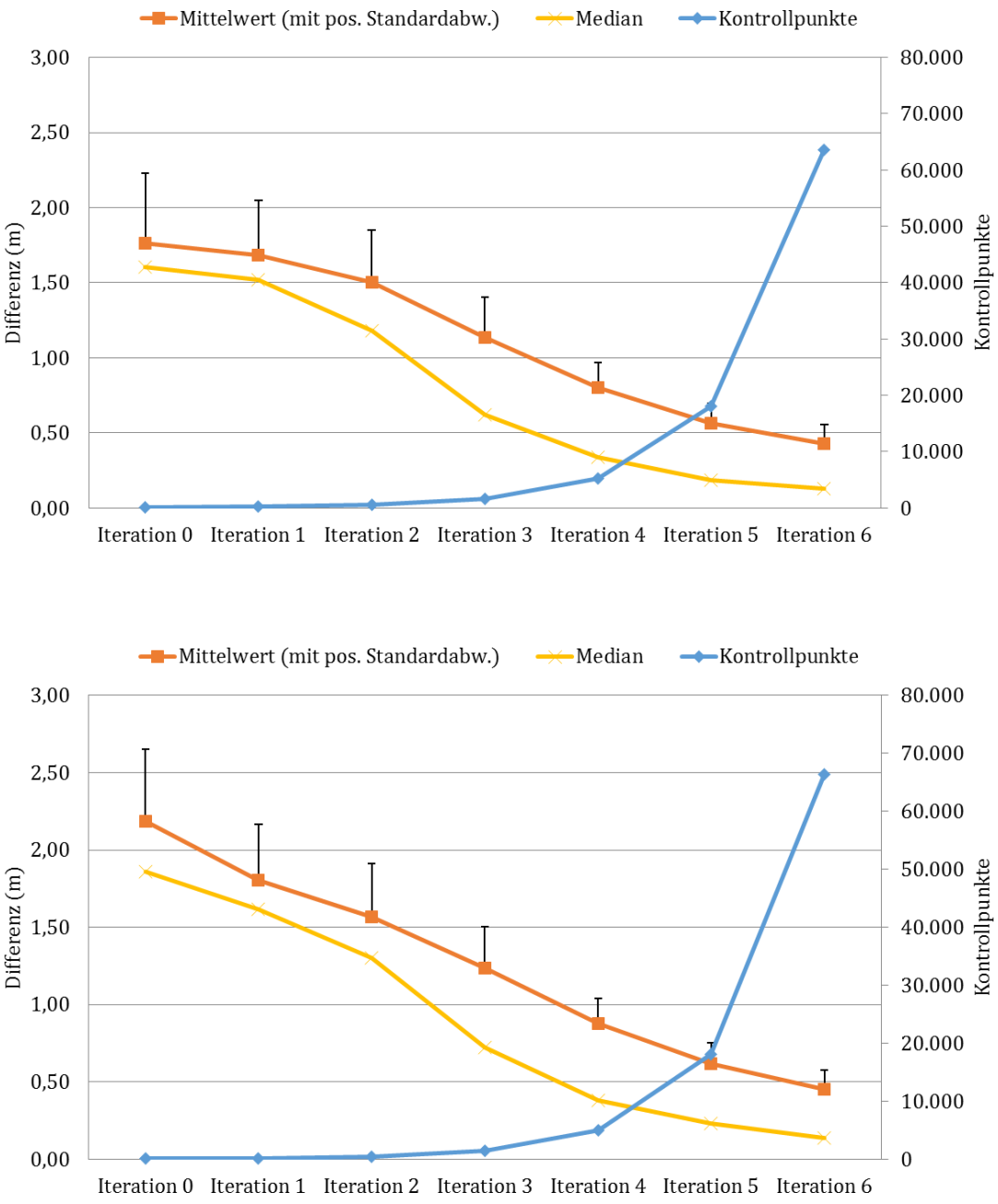


Abbildung 7.7: Vergleich der Qualität des Dragging-Verfahrens in der Bathymetrie (oben: linear, unten: kubisch)

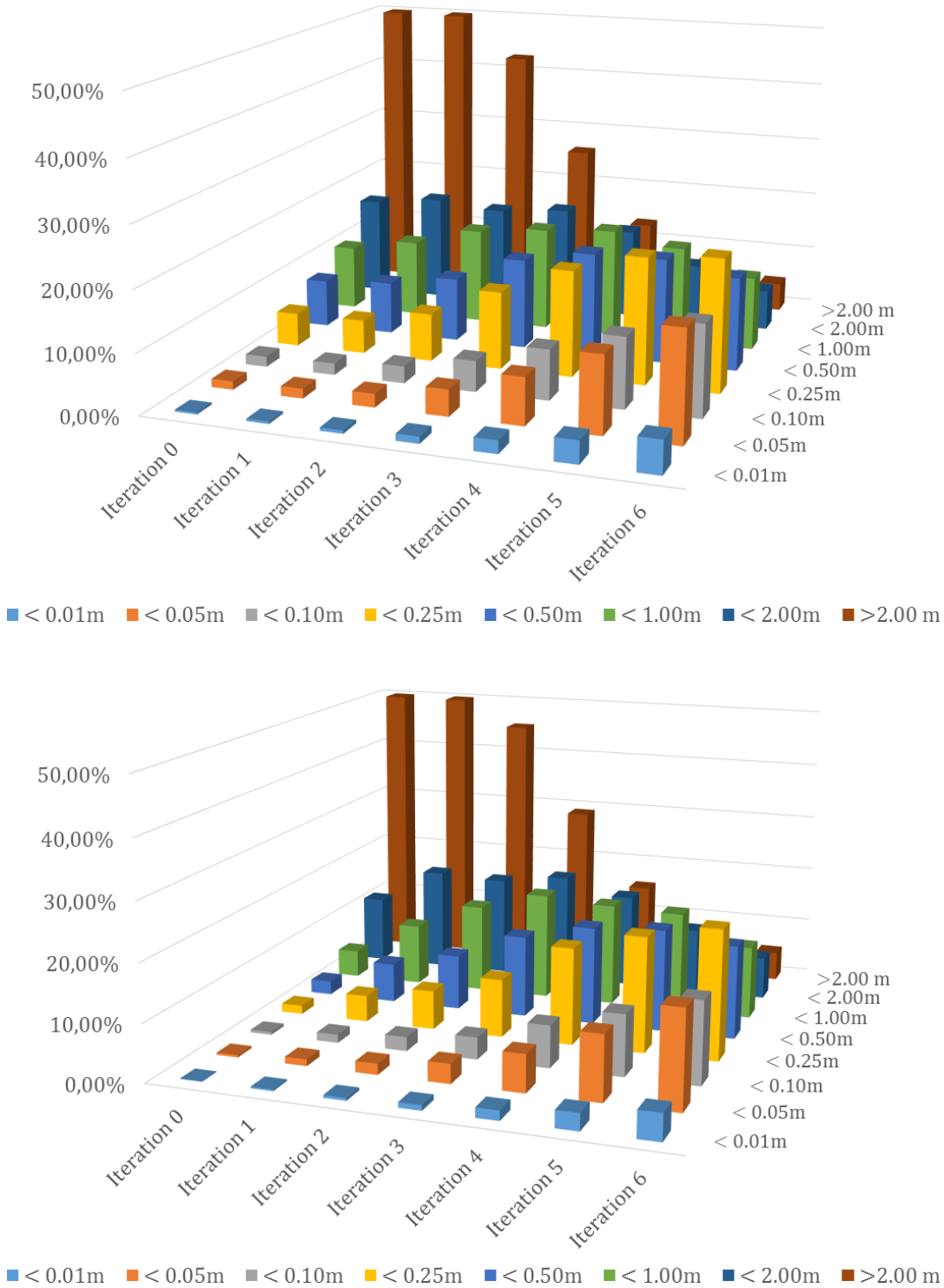


Abbildung 7.8: Histogramme des Dragging-Verfahrens

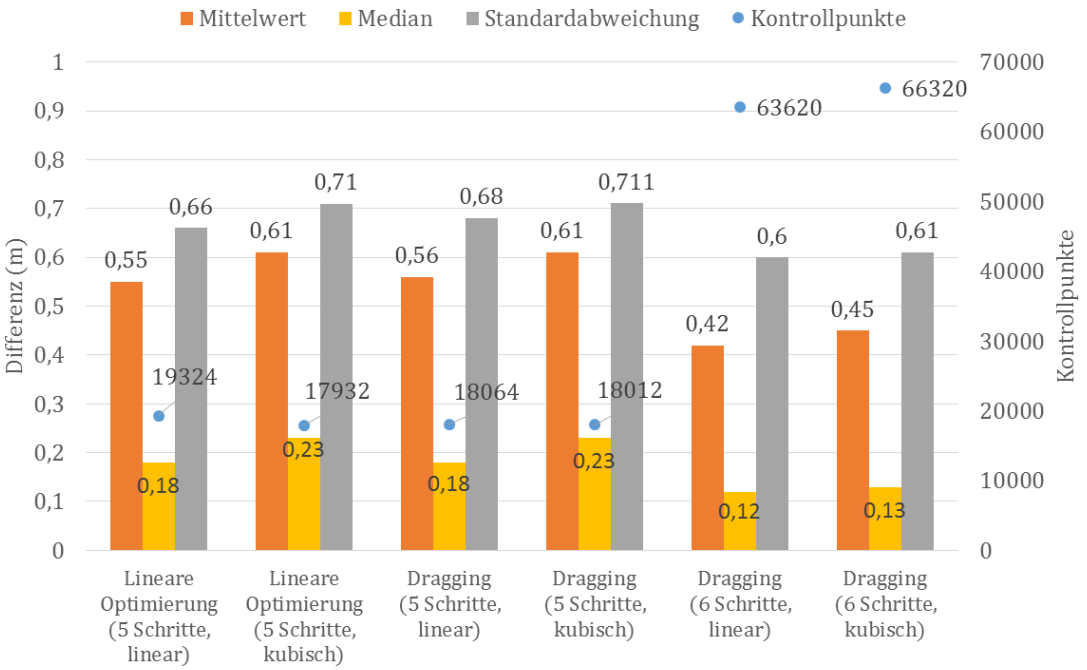


Abbildung 7.9: Vergleich der Optimierungsverfahren für ausgewählte Netzgrößen. Sichtbar sind der mittlere Differenzfehler, die positive Standardabweichung, der Median und die Anzahl der Kontrollpunkte.

7.2 Auswertung des Verfahrens

Es zeigt sich, dass beide Optimierungen grundsätzlich die erwartete Funktionalität aufweisen. Die erzeugten T-Netze und die damit verbundenen T-Spline-Flächen erfassen die originale Bathymetrie vollständig und bilden diese mit zunehmender Qualität ab. Mit dichteren T-Netzen wird die Bathymetrie besser erfasst (Abbildung 7.9). Die Ausdehnungen der Gebiete mit hohem Differenzfehler fallen mit zunehmender Stufe geringer aus. Starke Gradienten, d. h. stark abfallende Kanten wie der direkte Übergang in die Fahrrinne, sind ein offensichtliches Problem. Erwartungsgemäß liegt die Netzauflösung (d. h. die Abstände der Kontrollpunkte im T-Netz) deutlich oberhalb der Messpunktabstände. Je nach lokaler Dichte des T-Netzes liegen einzelne Kontrollpunkte mindestens 40 m auseinander, oftmals weitaus darüber. Eine detaillierte Erfassung von Unregelmäßigkeiten wird auf diesem Maßstab noch nicht erreicht, da hierfür deutlich dichtere Netze notwendig sind.

Die indirekte Reduktion verringert die Treppeneffekte in den T-Netzen bei diagonal verlaufenden Bathymetrien deutlich. Insgesamt deckt sie den Verlauf der Bathymetrie viel harmonischer ab, und die lokale Dichte der Kontrollpunkte variiert nur in wenigen Bereichen.

Der Vergleich der Optimierungen im betrachteten Beispiel spricht für das Dragging-Verfahren. Die Approximationsgüten fallen — bezogen auf die qualitativen Attribute der Differenzfehler — sowohl bei den erzeugten linearen als auch bei den kubischen T-Spline-Flächen vergleichbar gut aus. Mit dem Dragging-Verfahren können allerdings deutlich feinere T-Netze erzeugt werden. Hier zeigt sich der wesentliche Vorteil des Verfahrens gegenüber der linearen Optimierung und der damit verbundenen Abhängigkeit von den theoretischen und praktischen Beschränkungen der Lösungsverfahren. Auf der höchsten Stufe wird jedoch nicht die maximal mögliche Netzdichte erreicht und

die Messpunktabstände unterschreiten nicht die der Kontrollpunkte. Letztlich können somit nicht alle bathymetrischen Details, wie starke Gefälle an den Übergängen zu den Fahrrinnen oder extrem variable Gebiete, ausreichend gut erfasst werden.

In Bezug auf die Approximationsgüte zeigt sich eine Tendenz hin zu T-Spline-Flächen mit linearem Grad. Kubische Basisfunktionen erzeugen T-Spline-Flächen, deren Fehlergüten in der linearen Optimierung und im Dragging-Verfahren stets geringfügig höher liegen als lineare Basisfunktionen. Dieser Effekt lässt sich über die Glattheit der erzeugten T-Spline-Flächen bei höheren Graden erklären. Kubische (oder höhergradige) Kurven und Flächen verlaufen stets sanfter als ihre lineare Darstellung bei identischen Kontrollpunkten oder -netzen.

Ein allgemeiner Regelsatz für die Parametrisierung und Aussagen über die damit verbundenen Auswirkungen auf das Reduktionsverfahrens lassen sich aus den betrachteten Beispielen nicht gesichert ableiten. Generell sollten Anwender versuchen, die größtmöglichen Netze zu erzeugen und die semi-automatische Verfeinerung von T-Spline-Flächen (Abschnitt 4.1) derart zu kontrollieren, dass ein harmonisches T-Netz (im Sinne von möglichst vielen inneren Kontrollpunkten) entsteht. Hilfreich ist hierbei die Korrektur von hängenden Knoten bei Übergängen zu T-Netz-Zellen mit dichteren Nachbarschaften (vgl. Abschnitt 5.3). Derartige T-Netze stellen eine gleichmäßigere Verteilung der Kontrollpunkte sicher. Dies hat einen spürbaren Effekt auf die Qualität der linearen Optimierung mittels der Methode der kleinsten Quadrate.

7.3 Visualisierung

Die folgenden Abbildungen zeigen

- T-Netze für lineare T-Spline-Flächen mit der linearen Optimierung (Abbildungen 7.10 und 7.11).
- T-Netze für kubische T-Spline-Flächen mit der linearen Optimierung (Abbildungen 7.12 und 7.13).
- die erzeugten linearen T-Spline-Flächen nach Ablauf der linearen Optimierung für die Stufen 3 (Abbildung 7.14) bis 5 (Abbildung 7.16).
- die erzeugten kubischen T-Spline-Flächen nach Ablauf der linearen Optimierung für die Stufen 3 (Abbildung 7.17) bis 5 (Abbildung 7.19).
- T-Netze für lineare T-Spline-Flächen mit dem Dragging-Verfahren (Abbildungen 7.20 bis 7.22).
- T-Netze für kubische T-Spline-Flächen mit dem Dragging-Verfahren (Abbildungen 7.23 bis 7.25).
- die erzeugten linearen T-Spline-Flächen mit dem Dragging-Verfahren für die Stufen 3 (Abbildung 7.26) bis 6 (Abbildung 7.29).
- die erzeugten kubischen T-Spline-Flächen mit dem Dragging-Verfahren für die Stufen 3 (Abbildung 7.30) bis 6 (Abbildung 7.33).

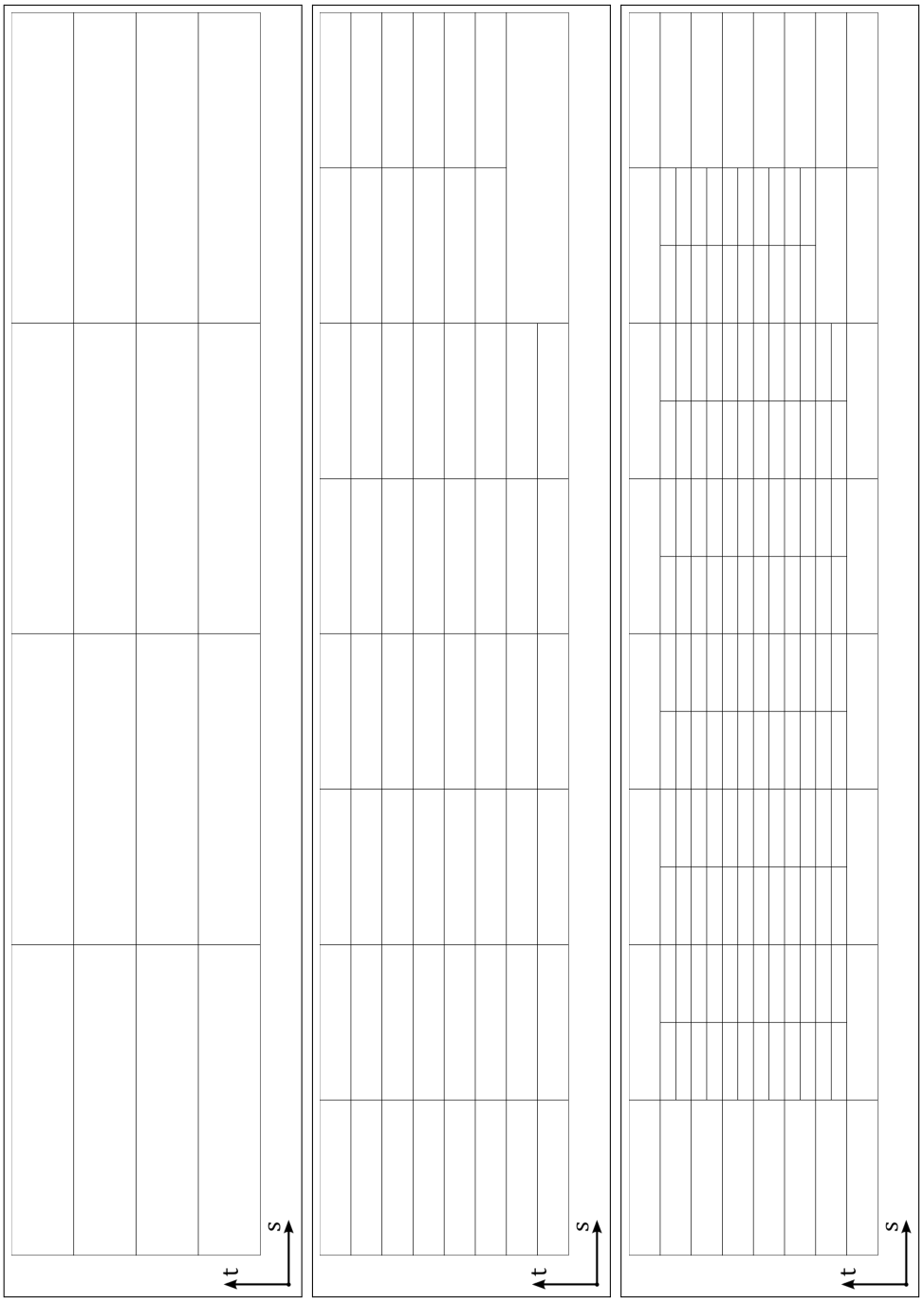


Abbildung 7.10: T-Netze bei linearer Optimierung für lineare T-Spline-Flächen (Stufe 0 bis 2)

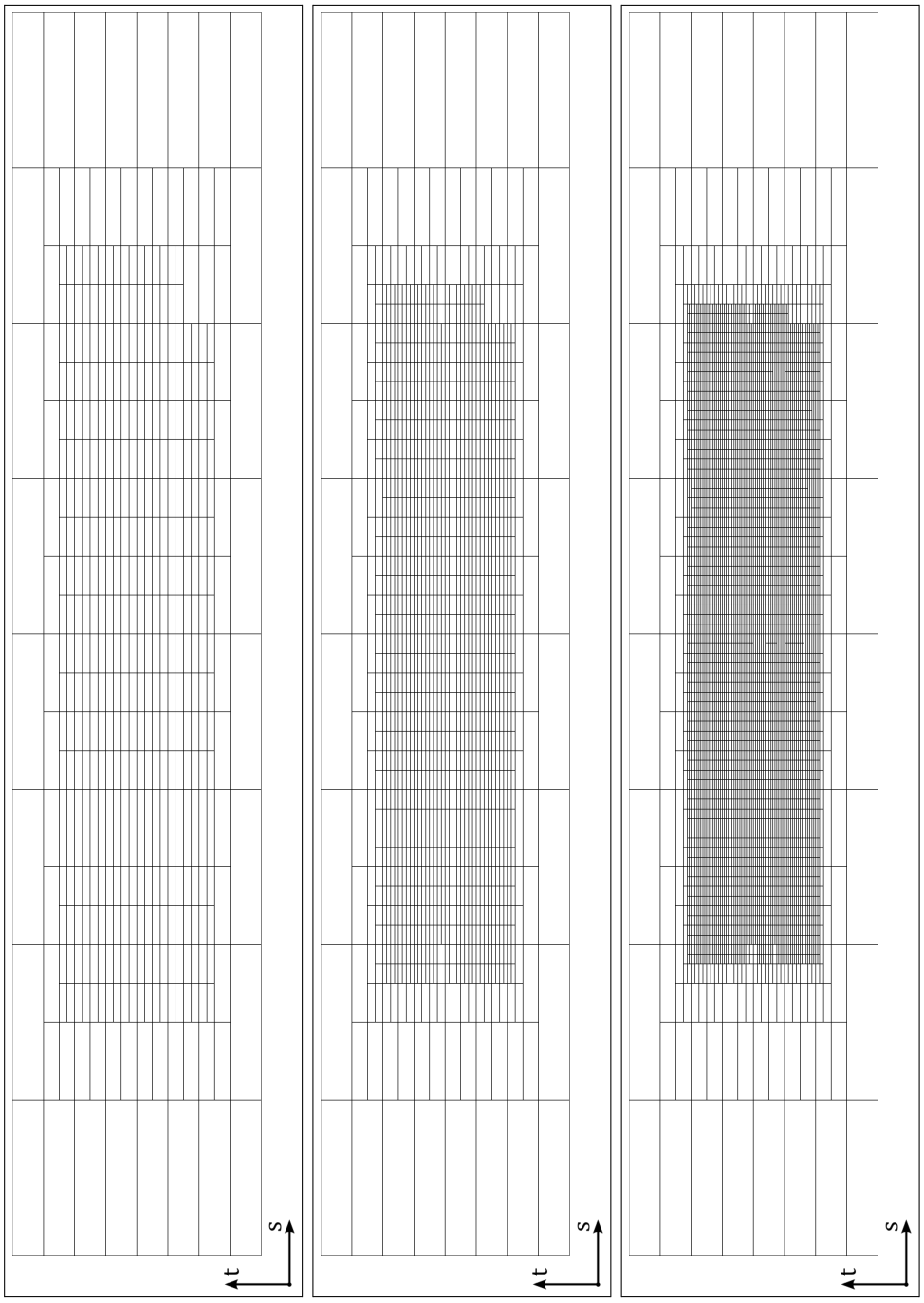


Abbildung 7.11: T-Netze bei linearer Optimierung für lineare T-Spline-Flächen (Stufe 3 bis 5)

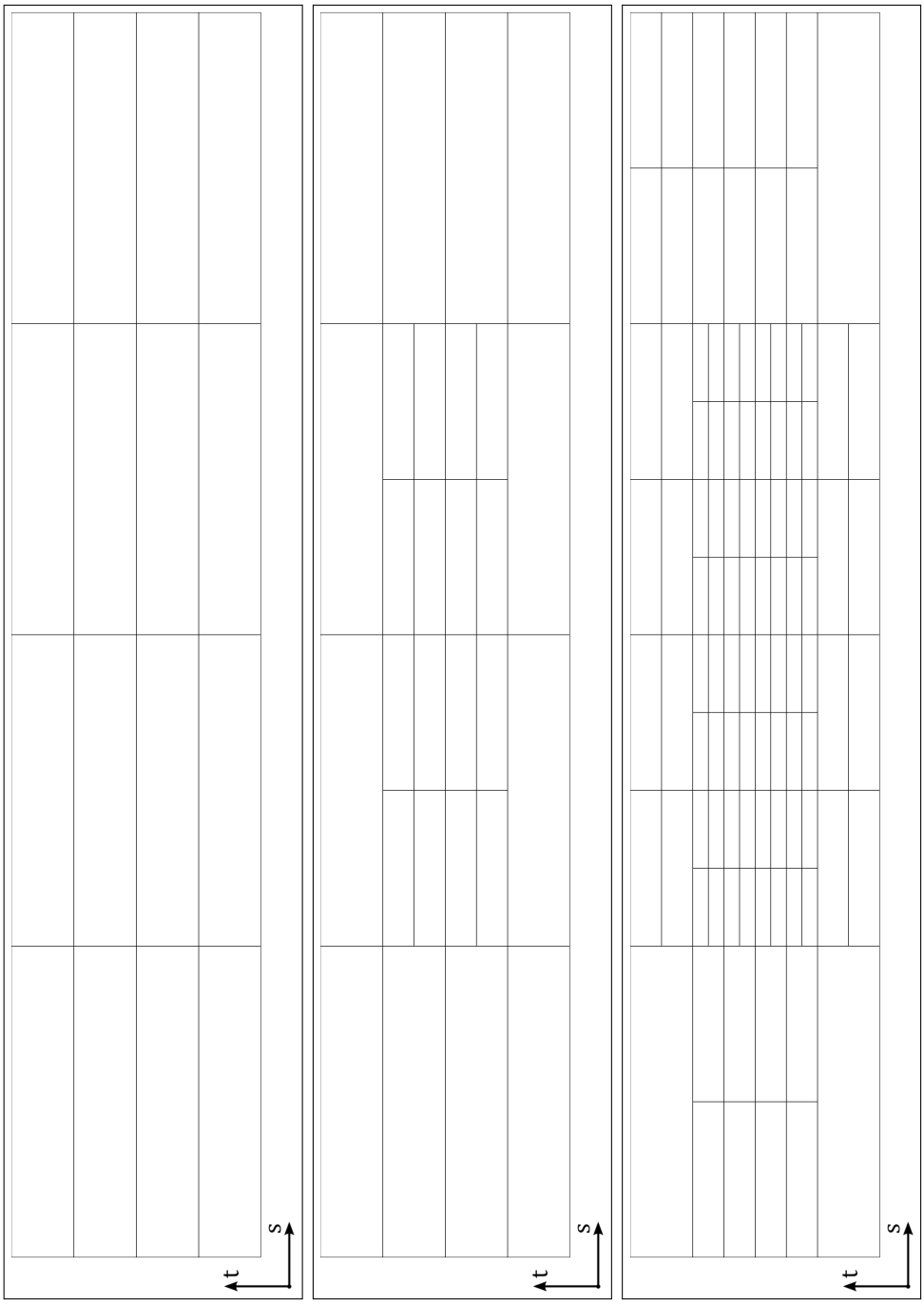


Abbildung 7.12: T-Netze bei linearer Optimierung für kubische T-Spline-Flächen (a)

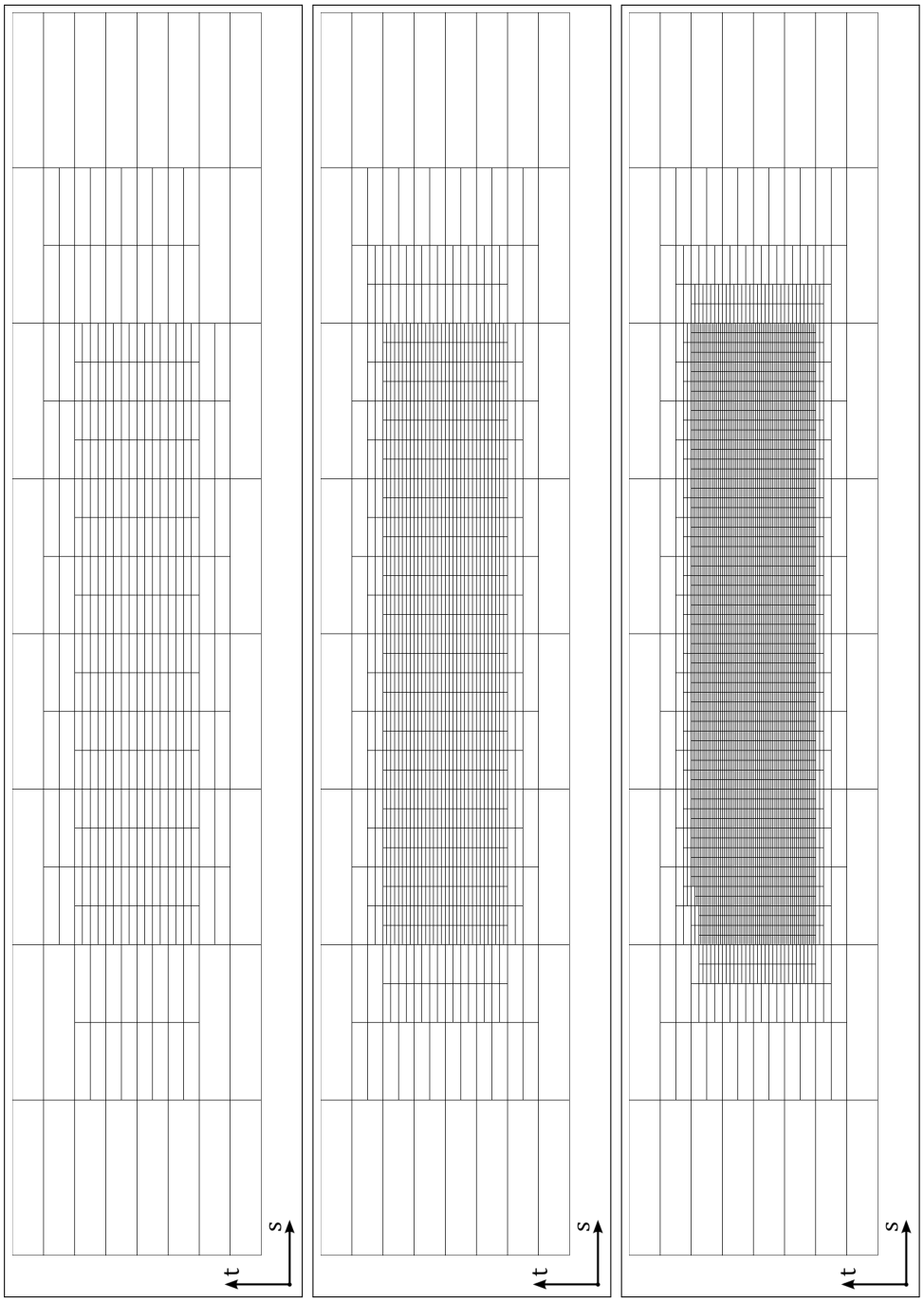


Abbildung 7.13: T-Netze bei linearer Optimierung für kubische T-Spline-Flächen (b)

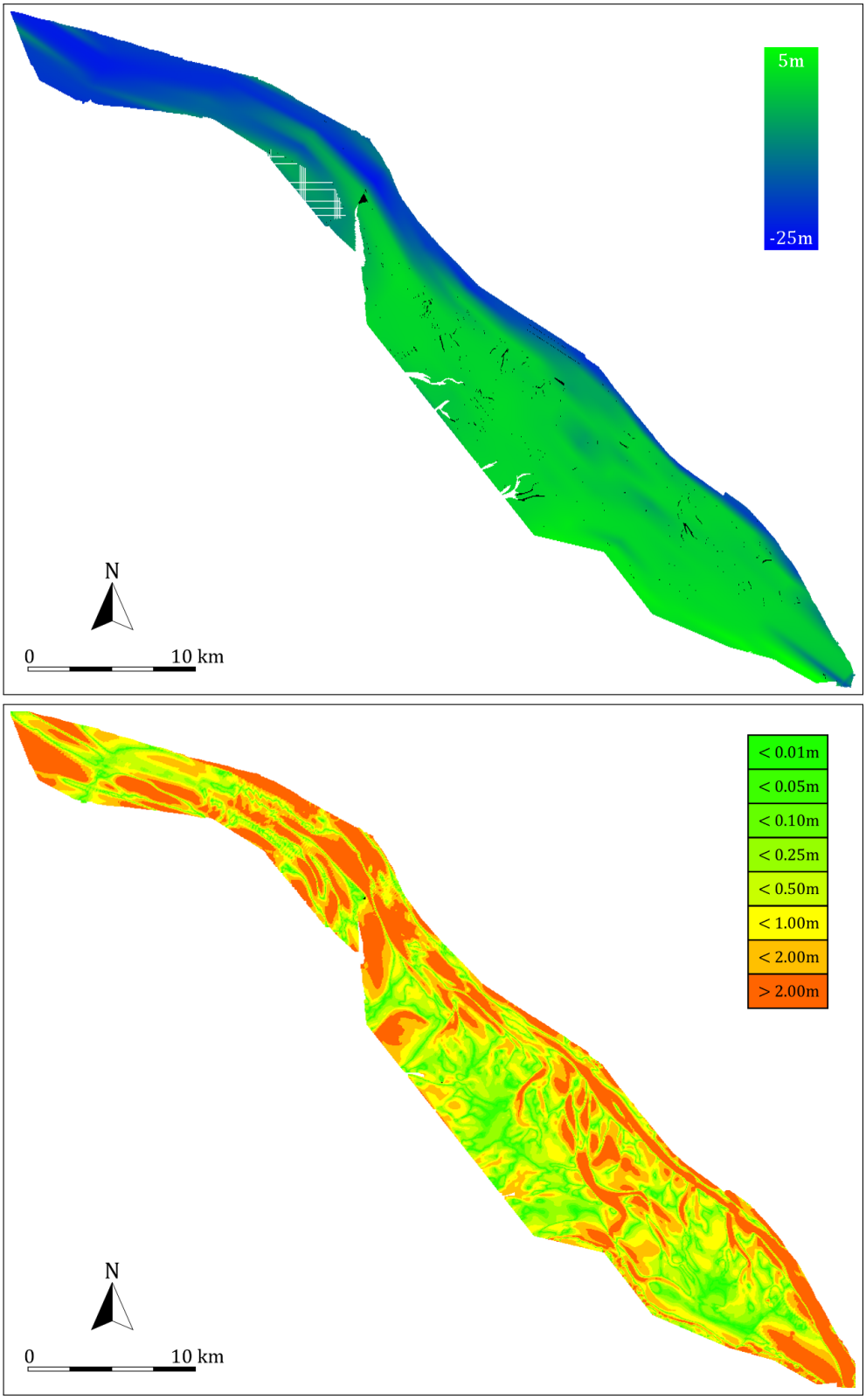


Abbildung 7.14: Lineare T-Spline-Fläche bei linearer Optimierung (3. Iteration). Dargestellt sind die Fläche (oben) von -25 m bis +5 m und die Klassen der Differenzfehler (unten).

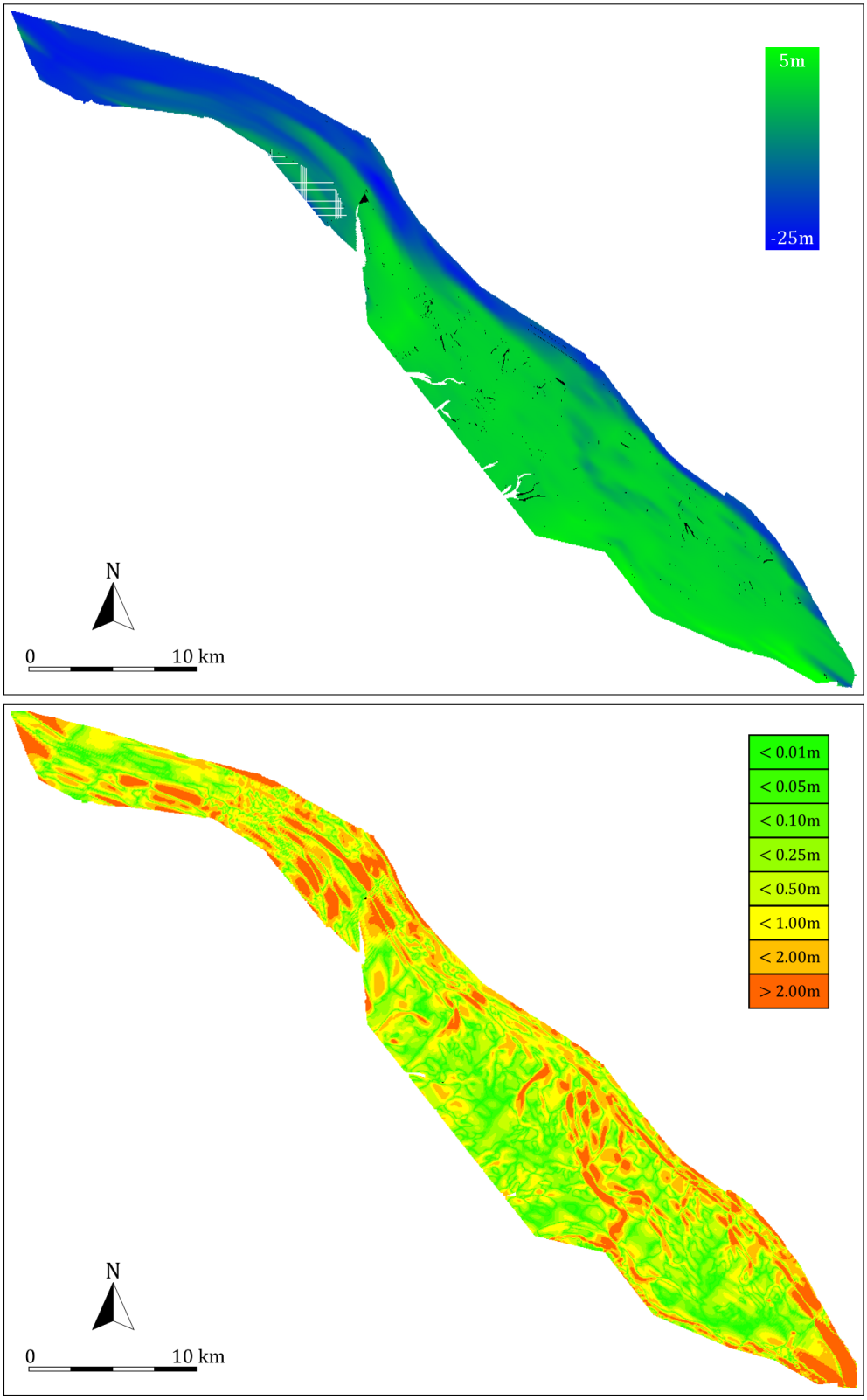


Abbildung 7.15: Lineare T-Spline-Fläche bei linearer Optimierung (4. Iteration)

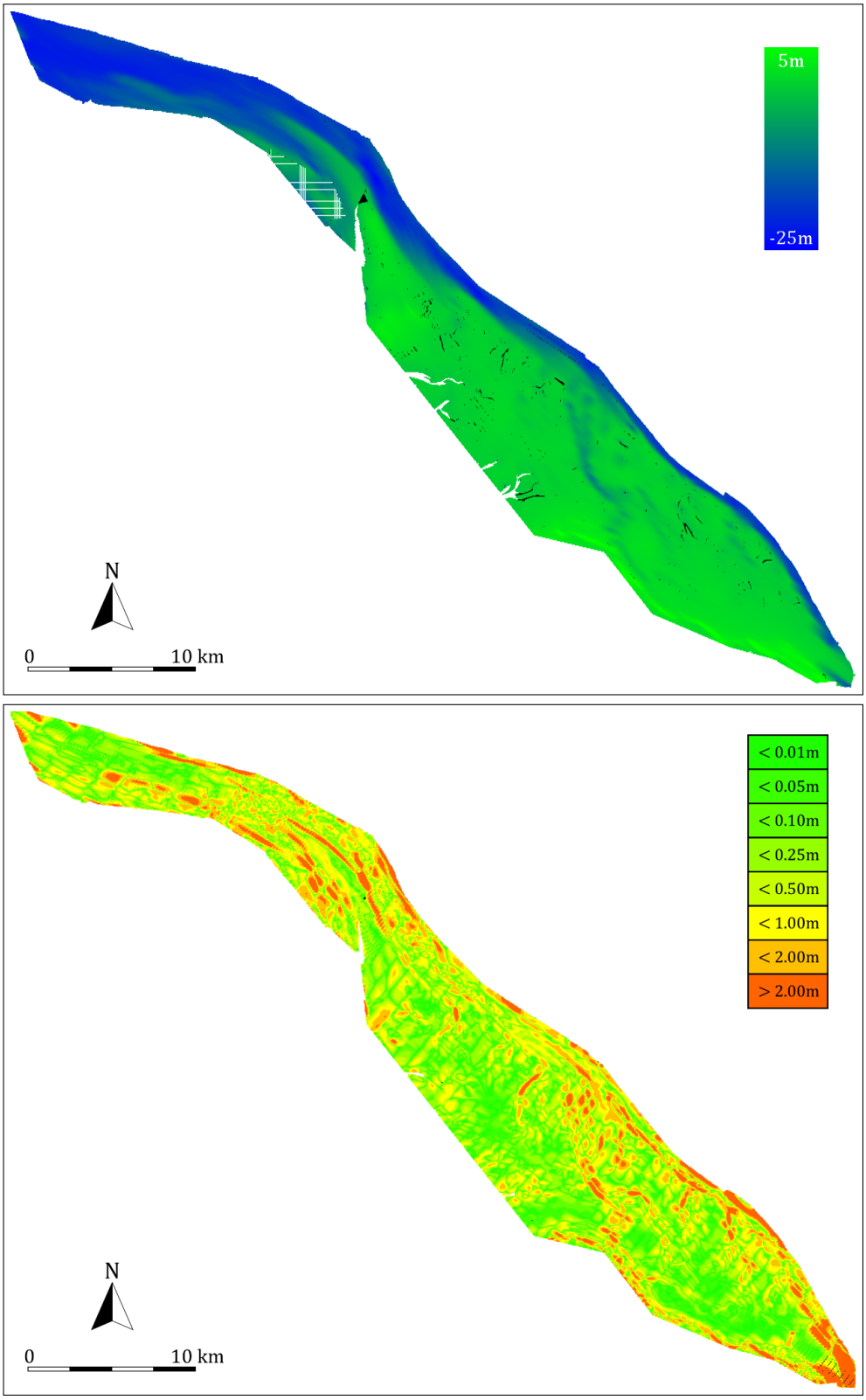


Abbildung 7.16: Lineare T-Spline-Fläche bei linearer Optimierung (5. Iteration)

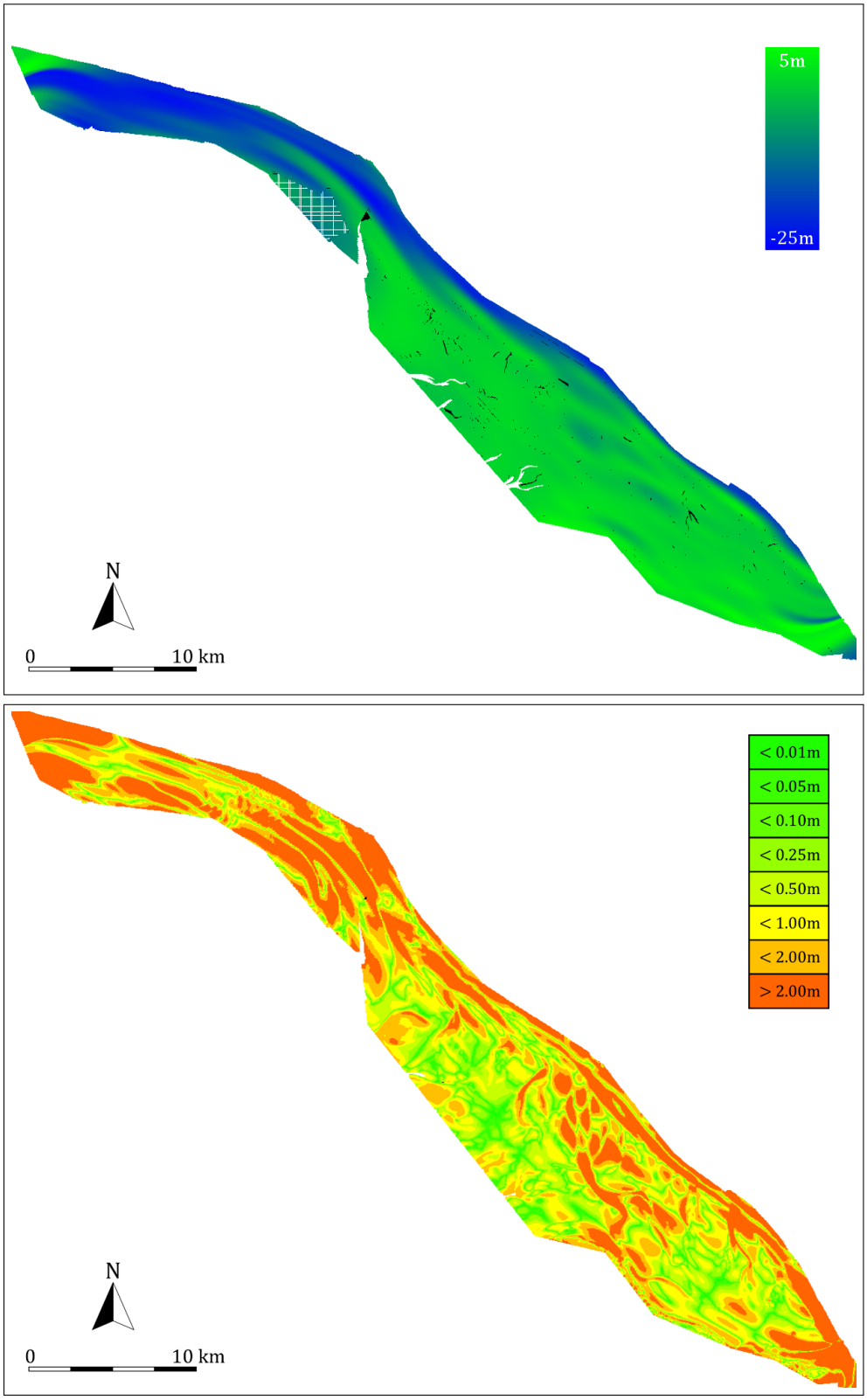


Abbildung 7.17: Kubische T-Spline-Fläche bei linearer Optimierung (3. Iteration)

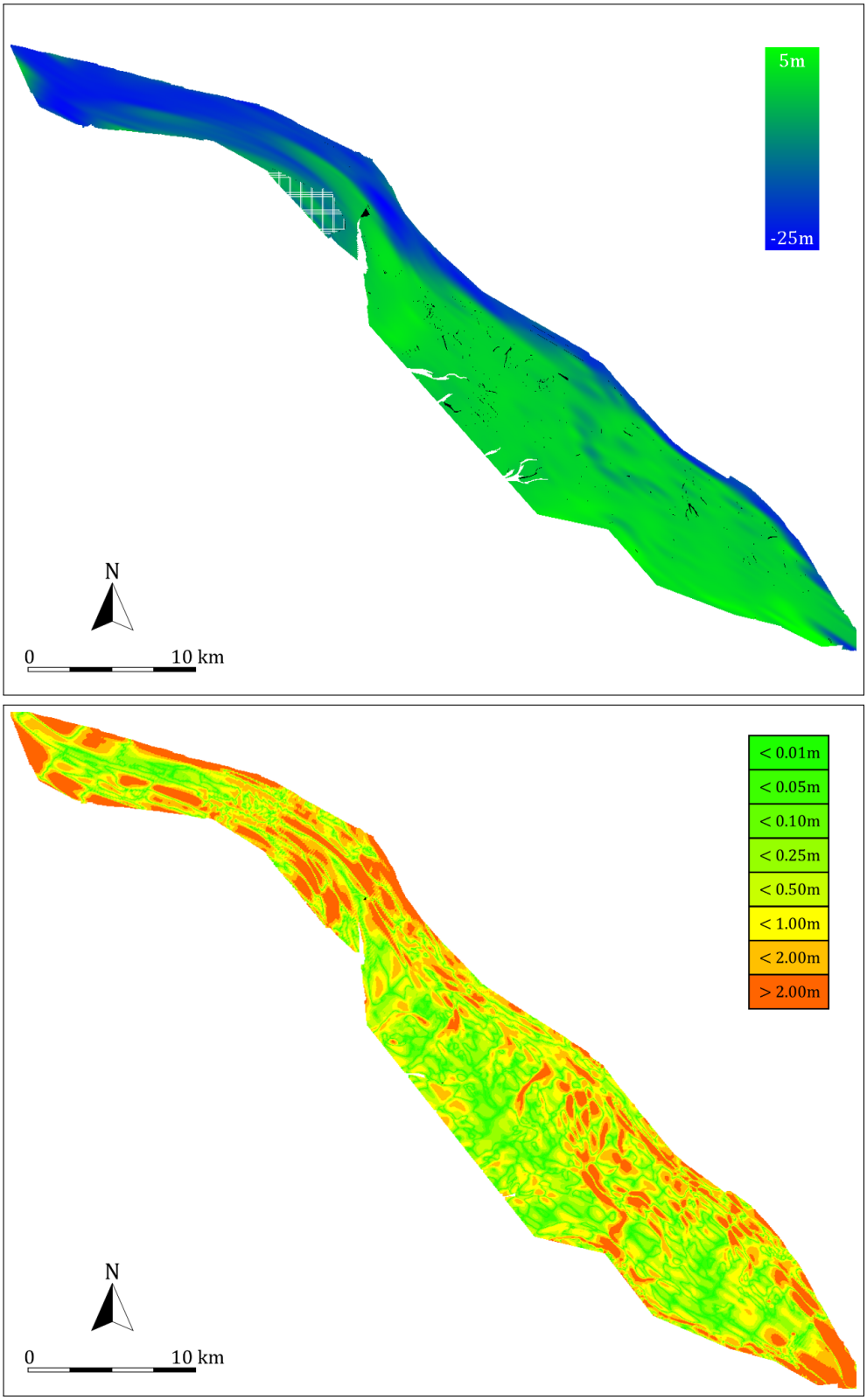


Abbildung 7.18: Kubische T-Spline-Fläche bei linearer Optimierung (4. Iteration)

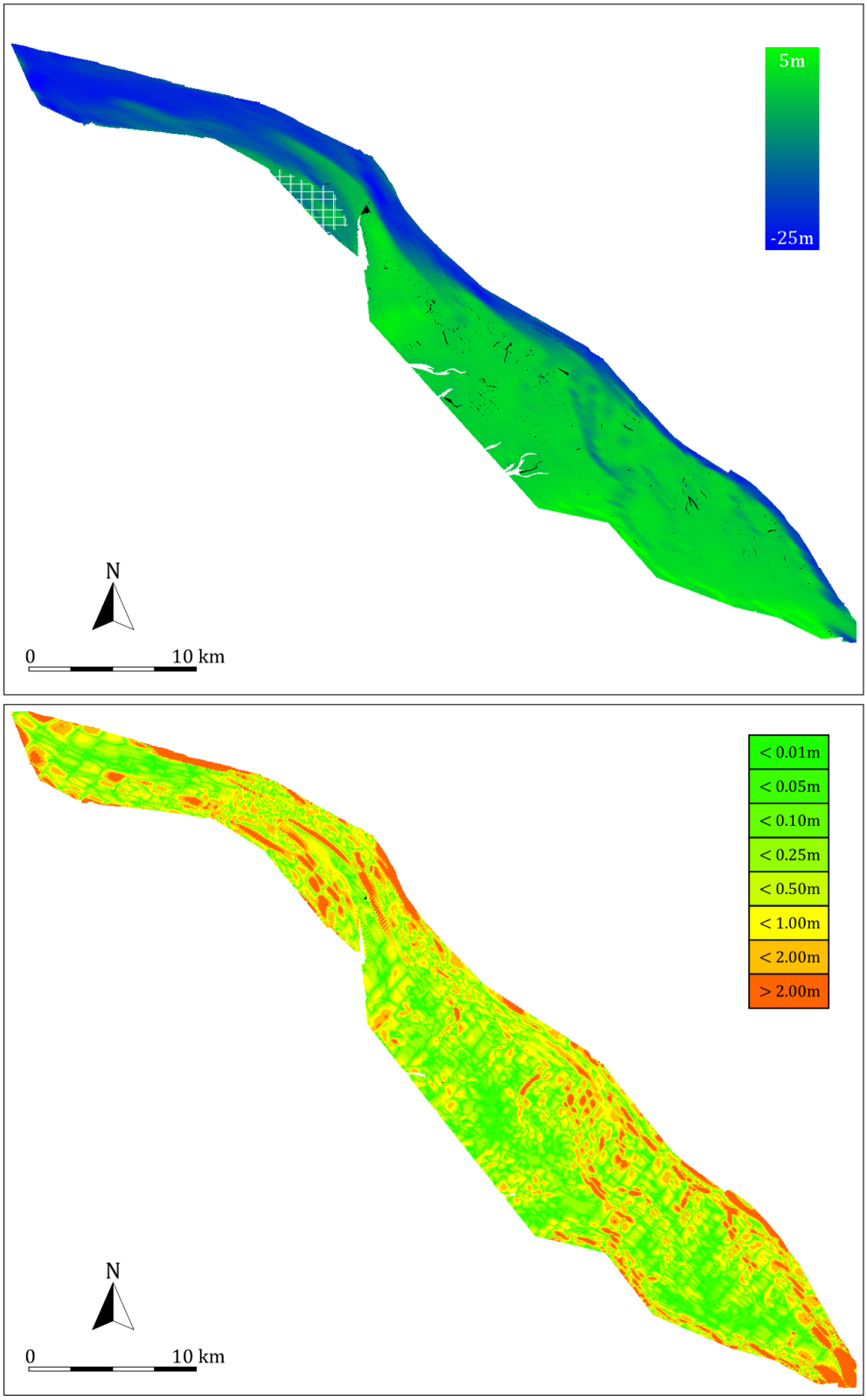


Abbildung 7.19: Kubische T-Spline-Fläche bei linearer Optimierung (5. Iteration)

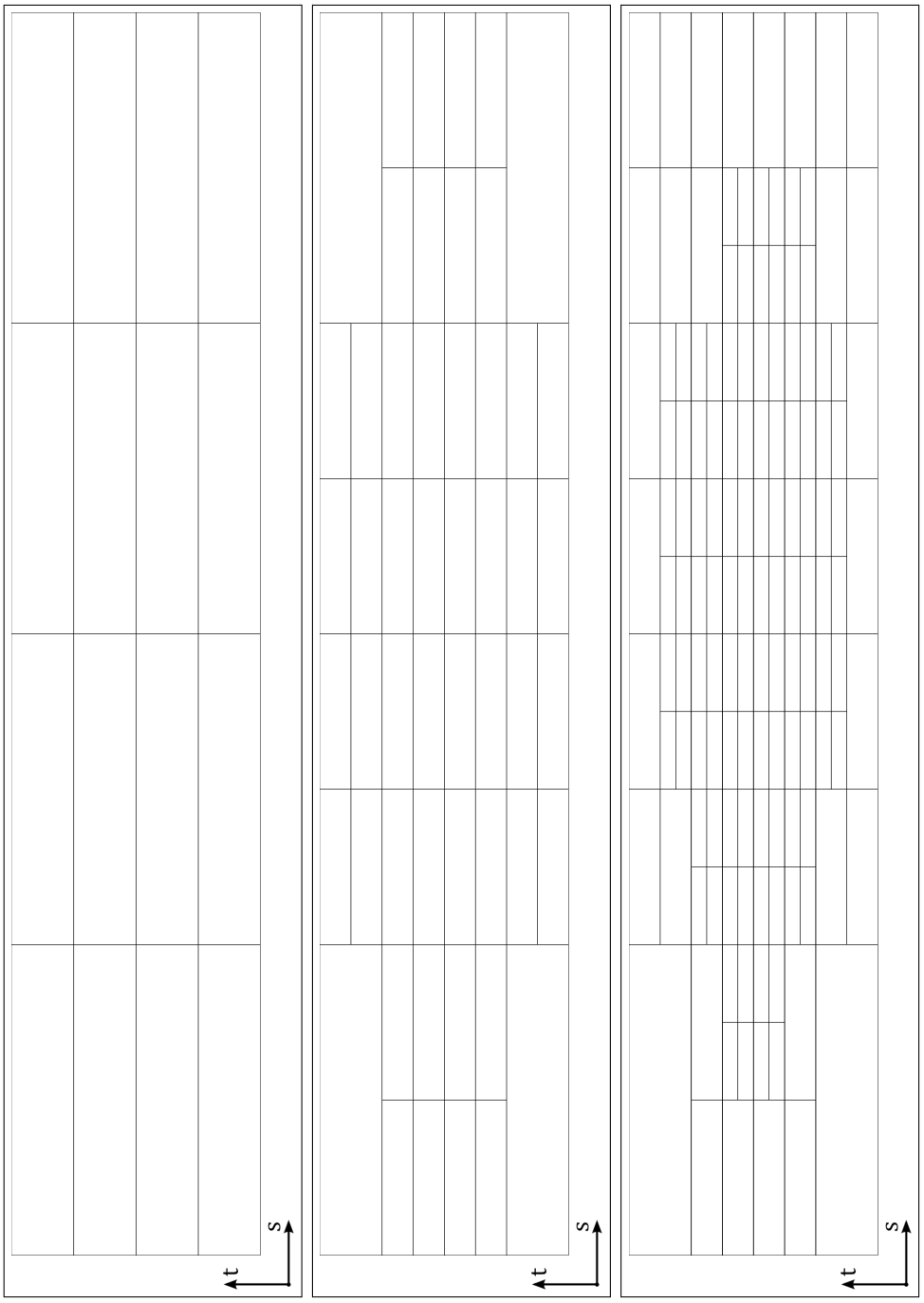


Abbildung 7.20: T-Netze mit dem Dragging-Verfahren für lineare T-Spline-Flächen (Stufe 0 bis 2)

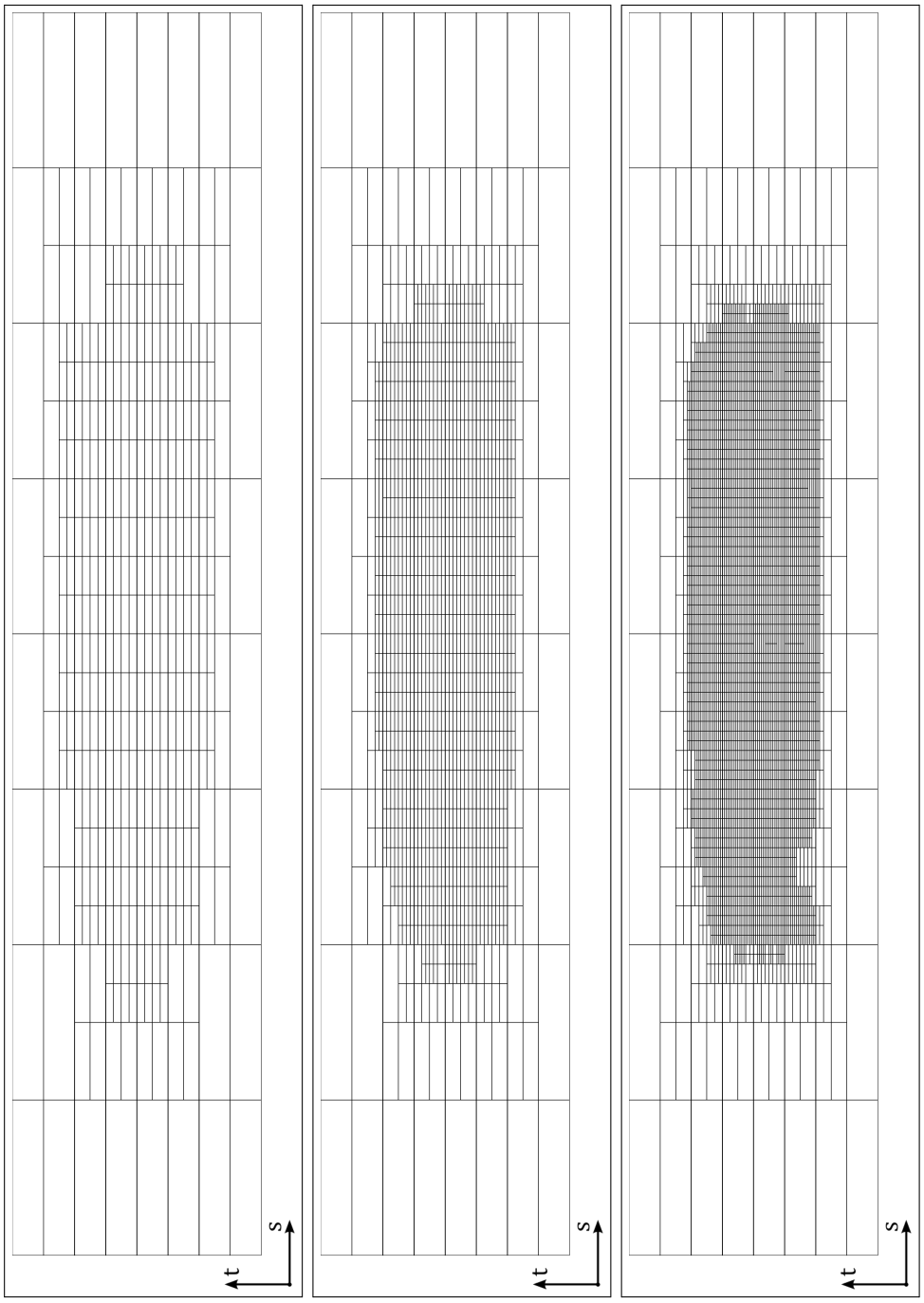


Abbildung 7.21: T-Netze mit dem Dragging-Verfahren für lineare T-Spline-Flächen (Stufe 3 bis 5)

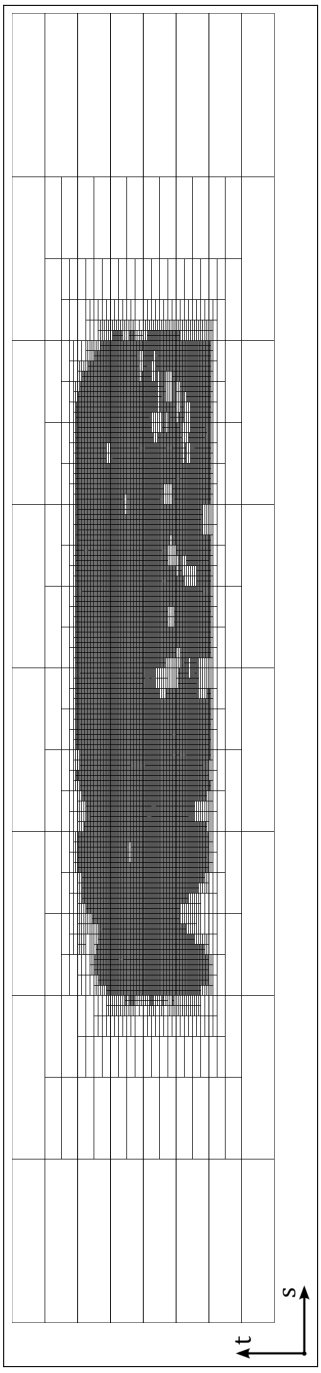


Abbildung 7.22: T-Netze mit dem Dragging-Verfahren für lineare T-Spline-Flächen (Stufe 6)

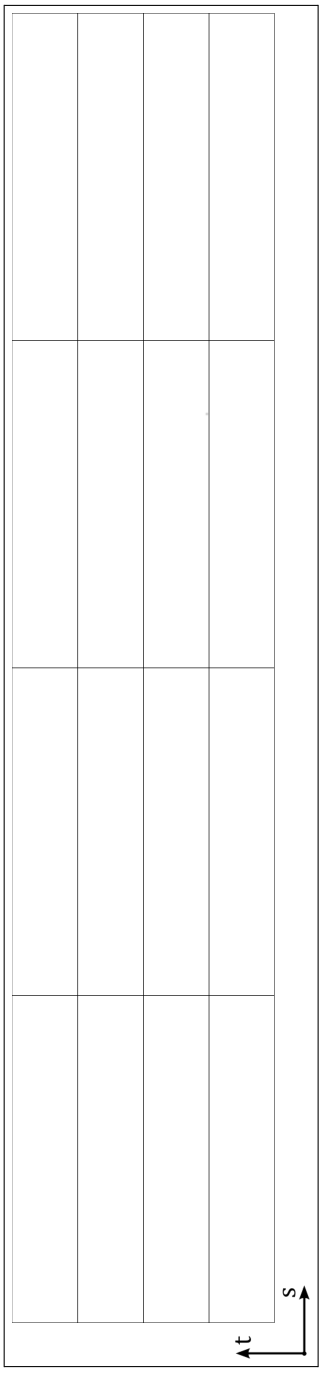


Abbildung 7.23: T-Netze mit dem Dragging-Verfahren für kubische T-Spline-Flächen (Stufe 0)

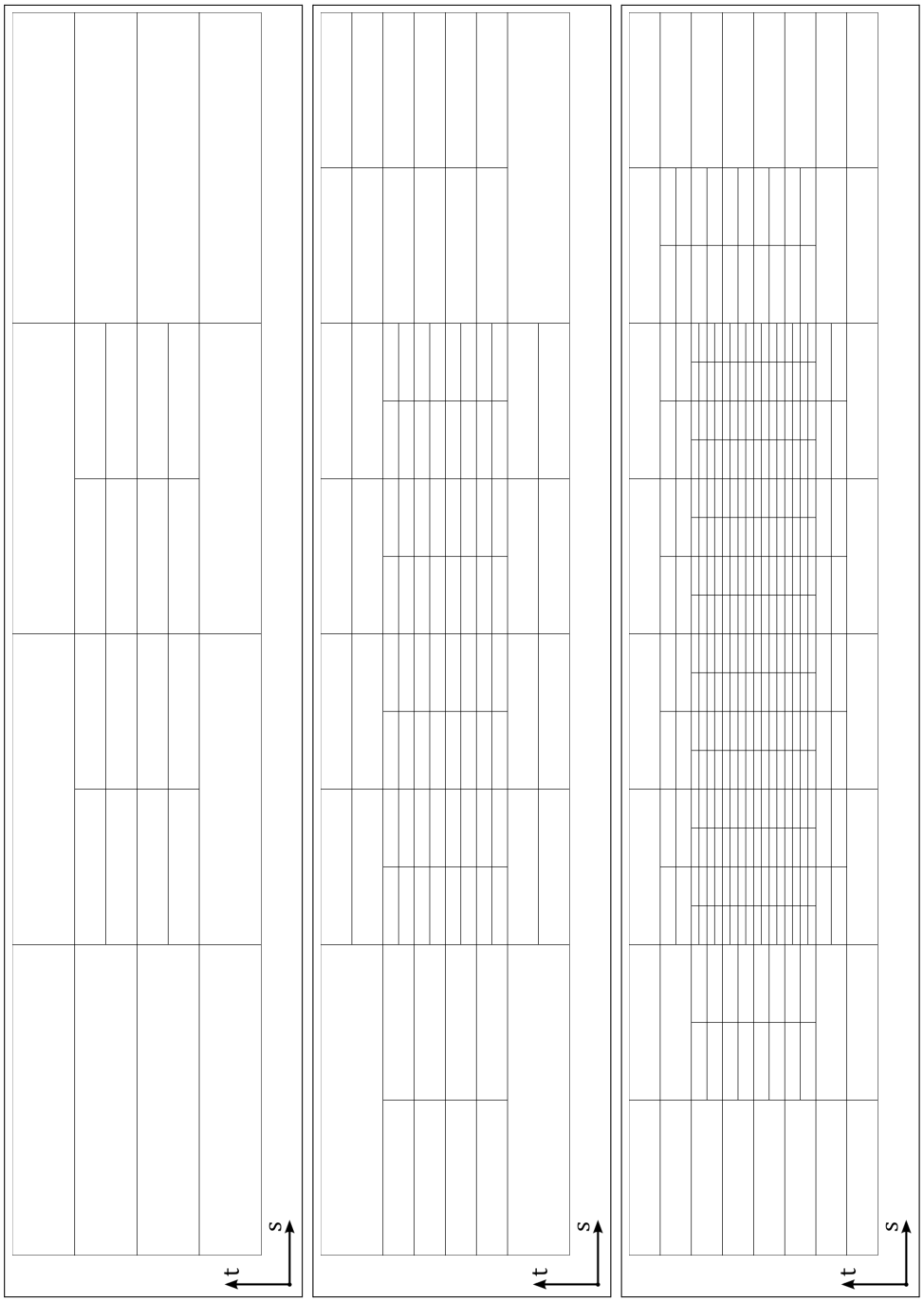


Abbildung 7.24: T-Netze mit dem Dragging-Verfahren für kubische T-Spline-Flächen (Stufe 1 bis 3)

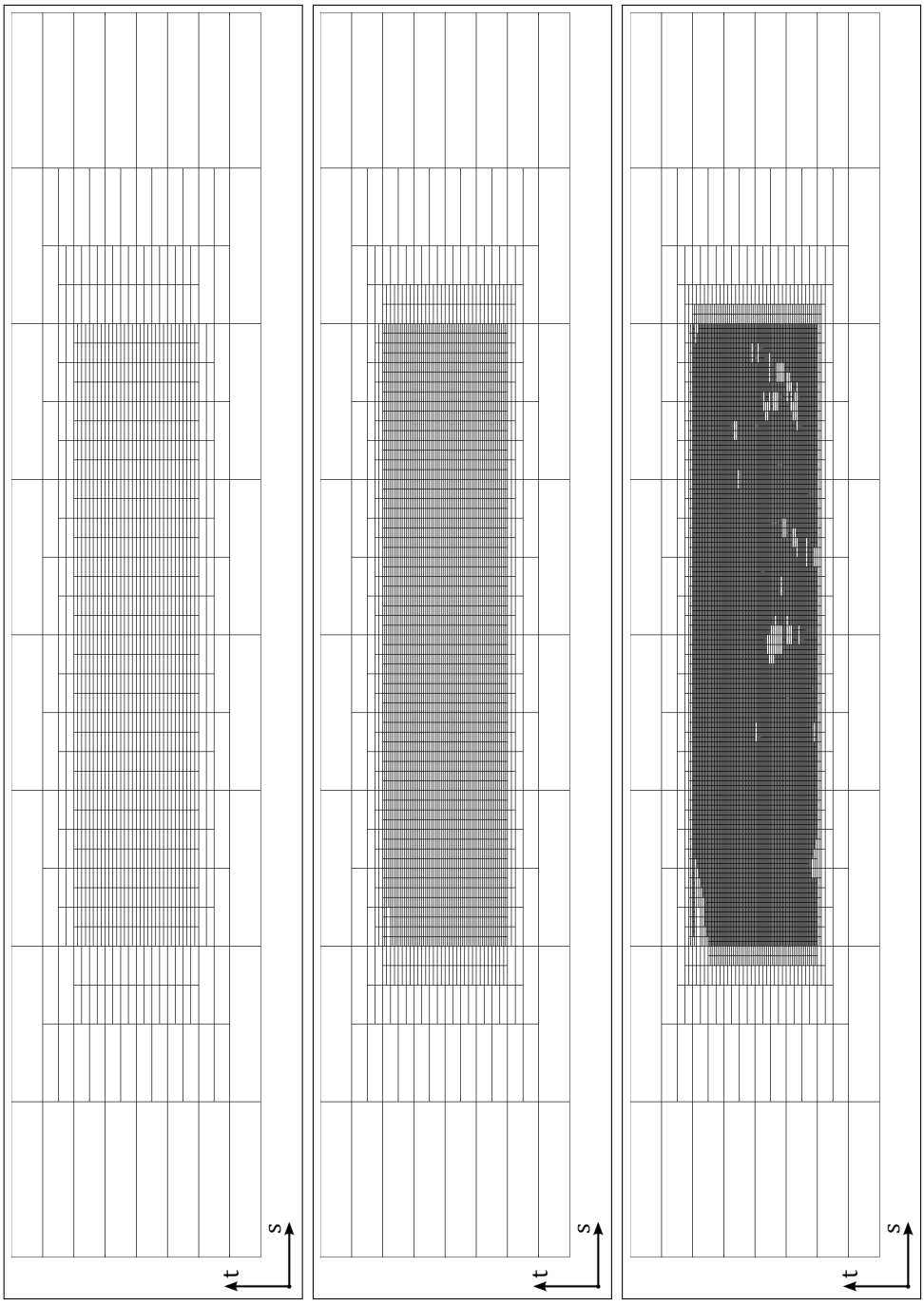


Abbildung 7.25: T-Netze mit dem Dragging-Verfahren für kubische T-Spline-Flächen (Stufe 4 bis 6)

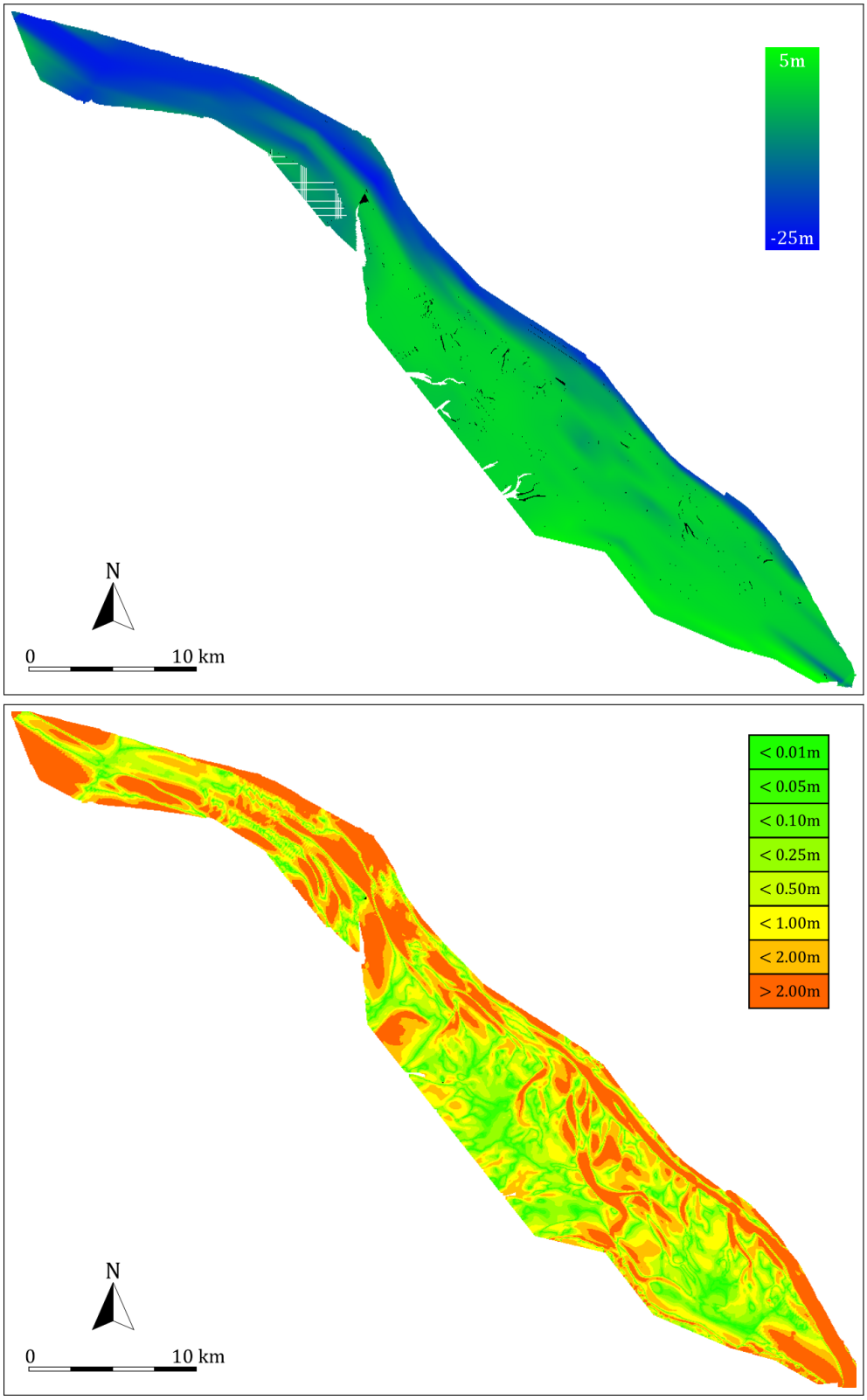


Abbildung 7.26: Lineare T-Spline-Fläche mit dem Dragging-Verfahren (3. Iteration)

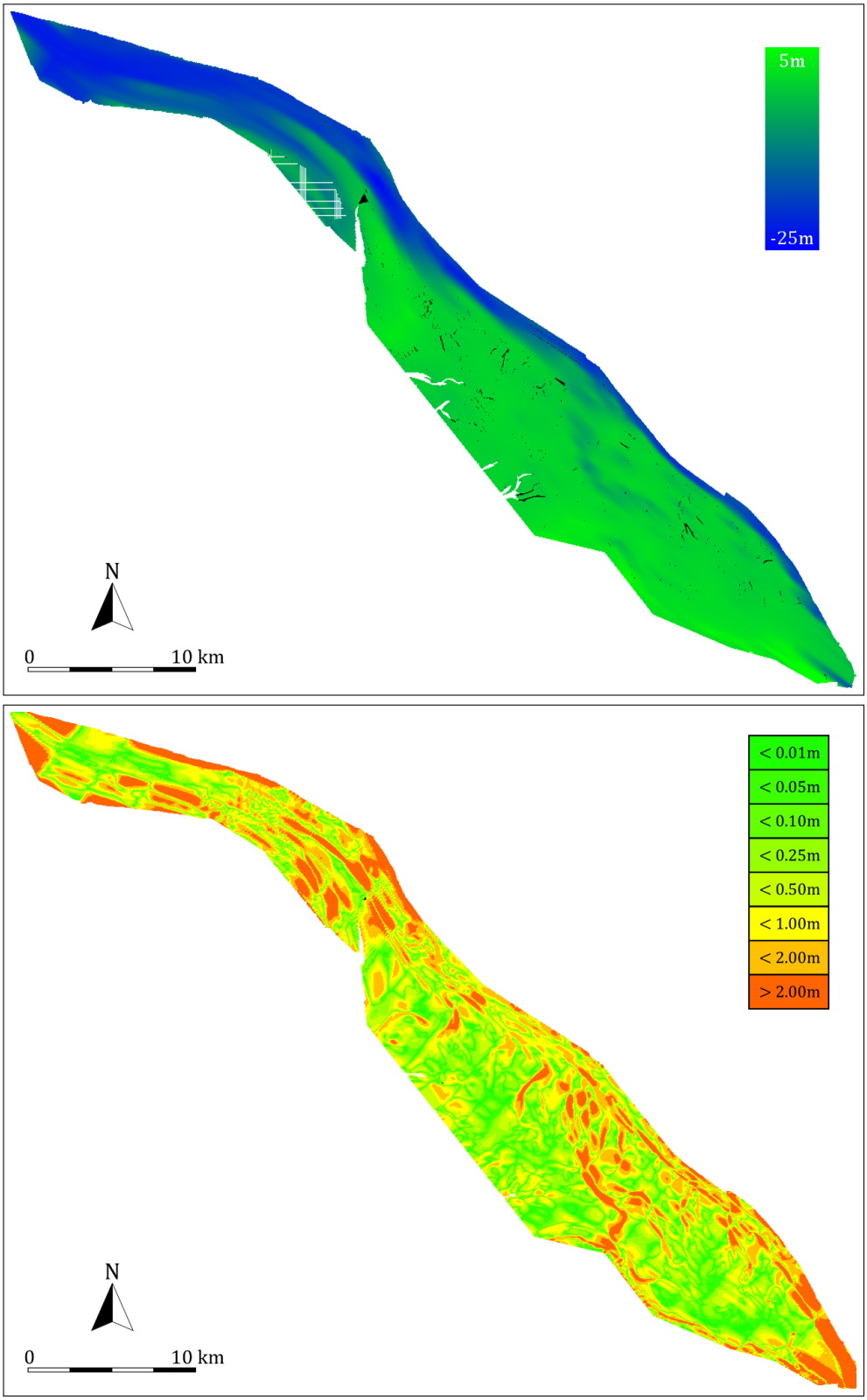


Abbildung 7.27: Lineare T-Spline-Fläche mit dem Dragging-Verfahren (4. Iteration)

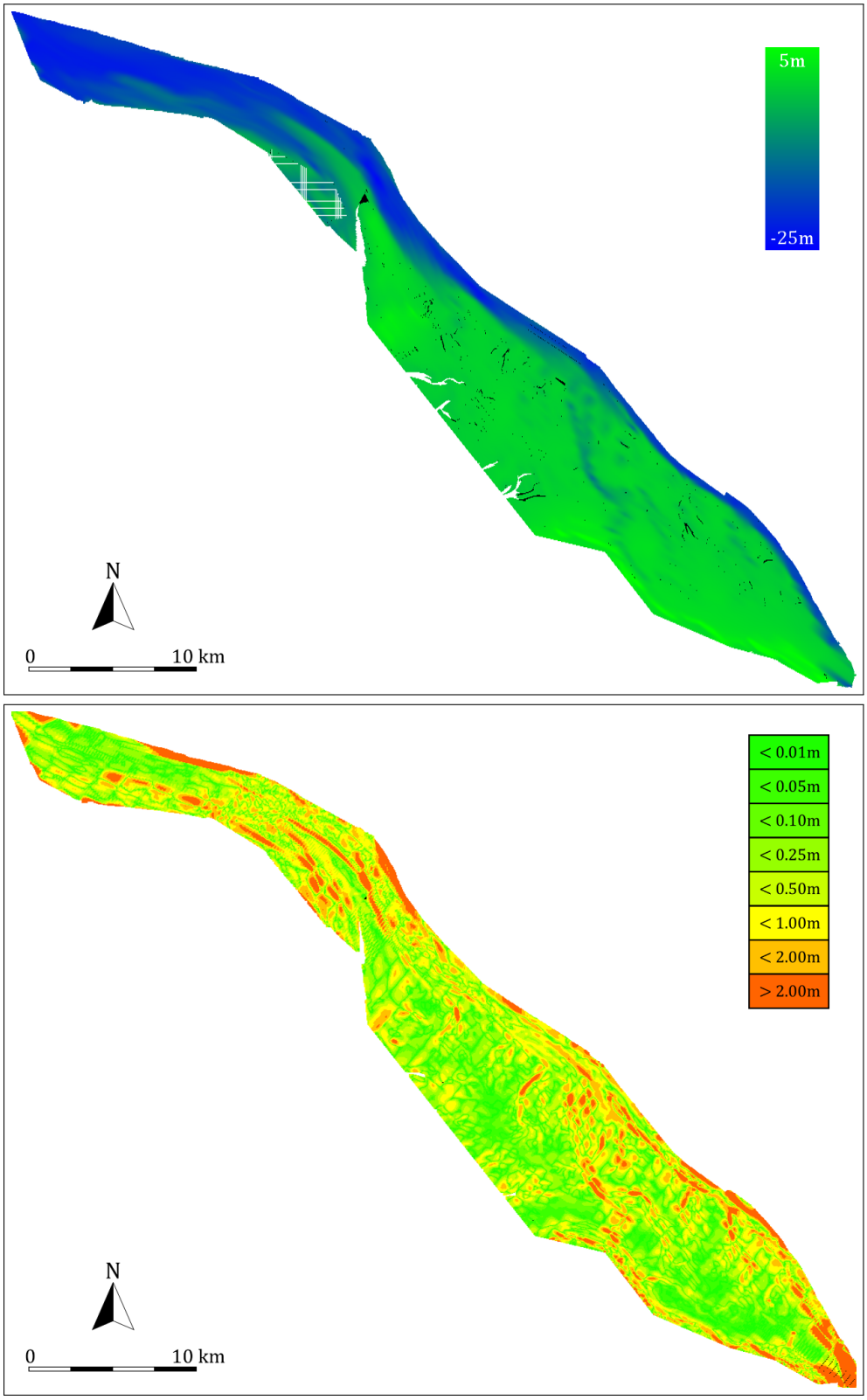


Abbildung 7.28: Lineare T-Spline-Fläche mit dem Dragging-Verfahren (5. Iteration)

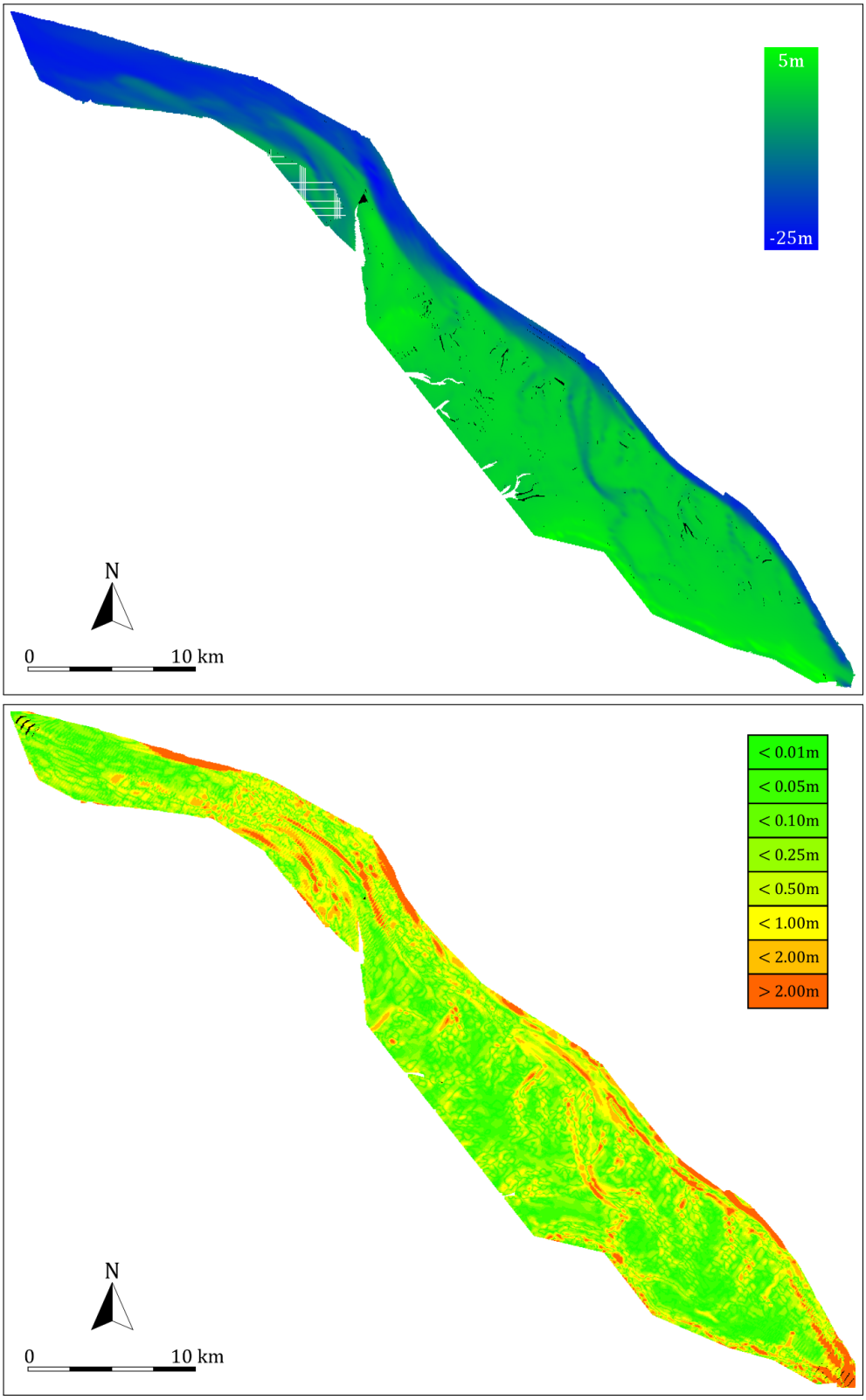


Abbildung 7.29: Lineare T-Spline-Fläche mit dem Dragging-Verfahren (6. Iteration)

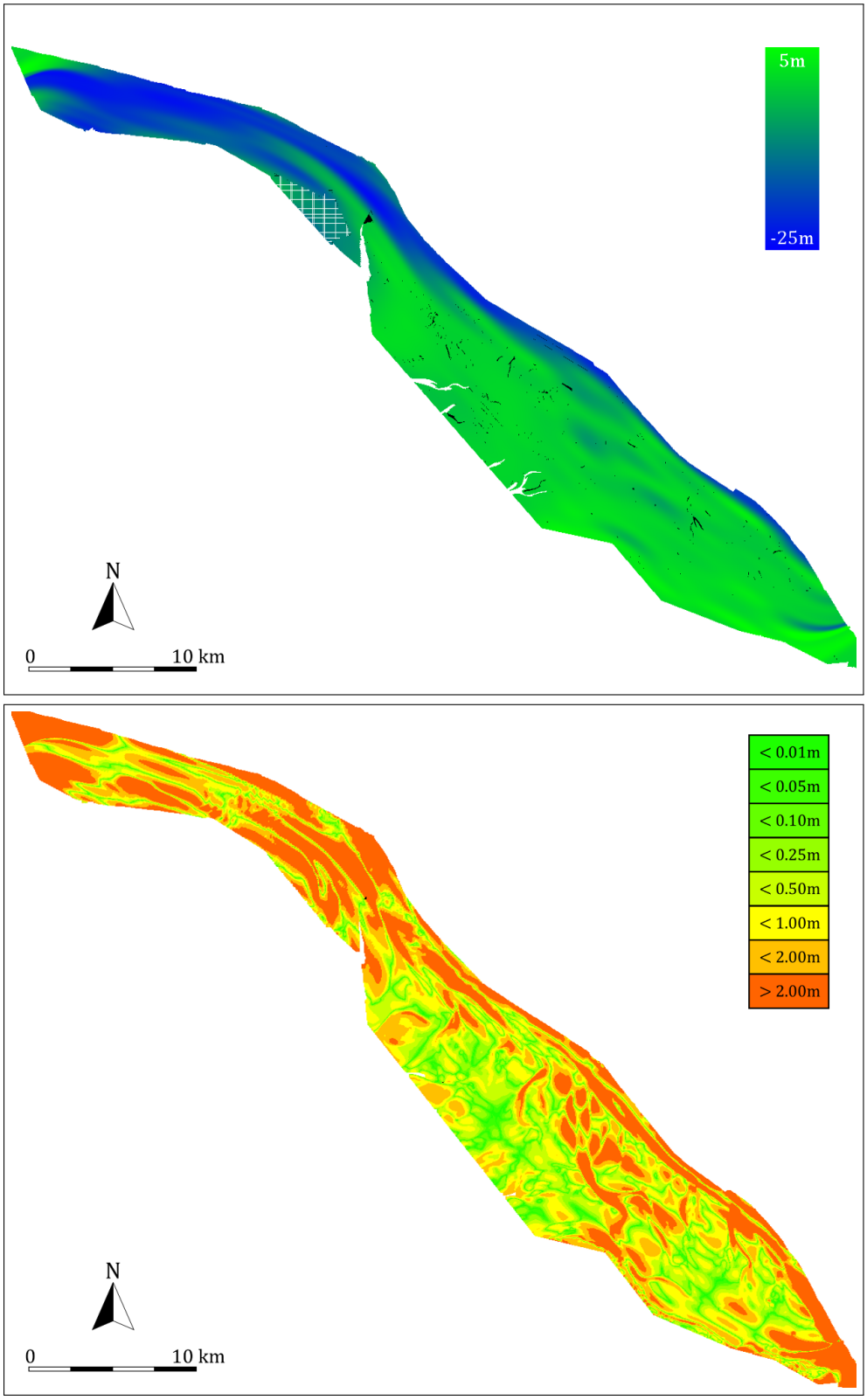


Abbildung 7.30: Kubische T-Spline-Fläche mit dem Dragging-Verfahren (3. Iteration)

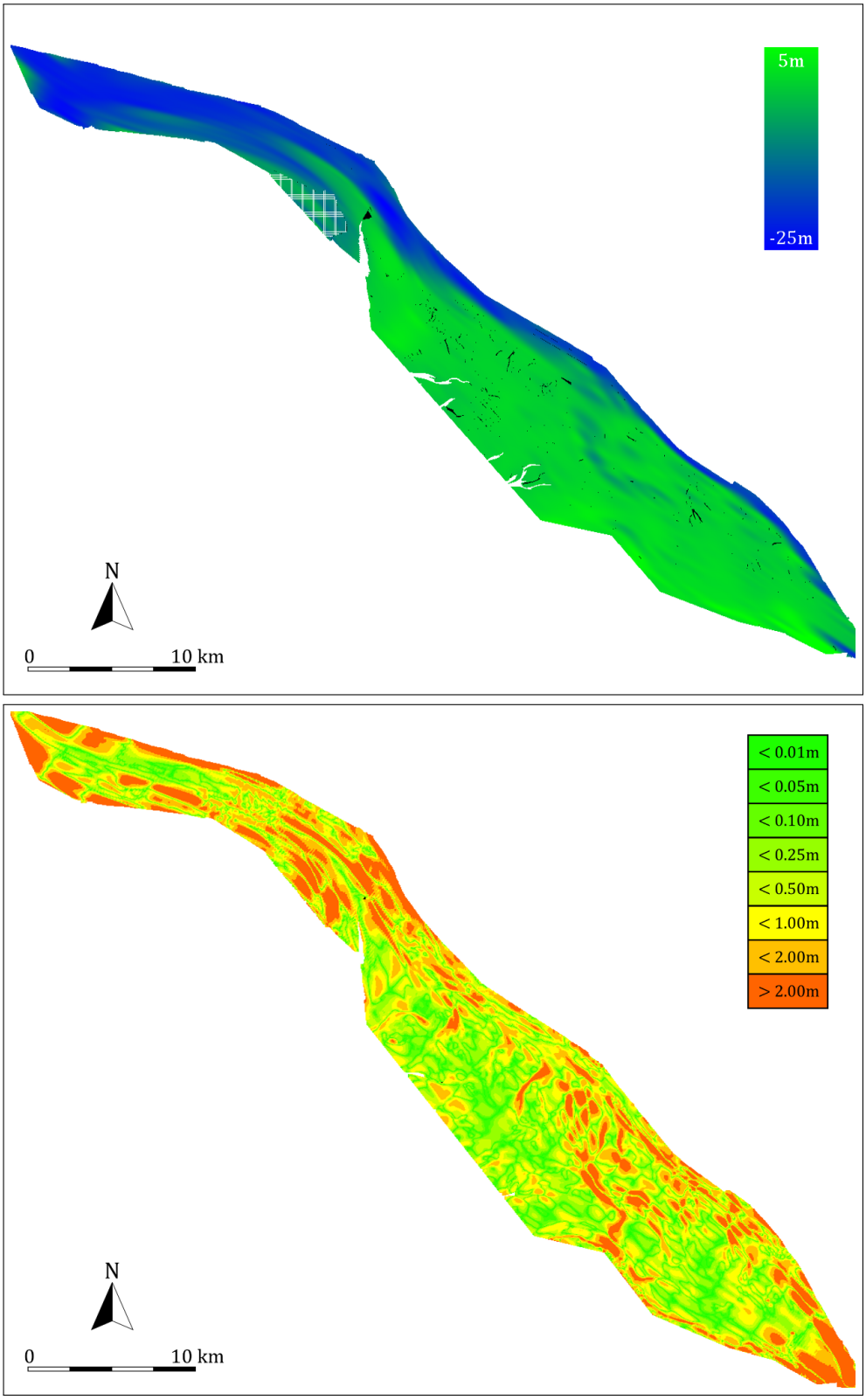


Abbildung 7.31: Kubische T-Spline-Fläche mit dem Dragging-Verfahren (4. Iteration)

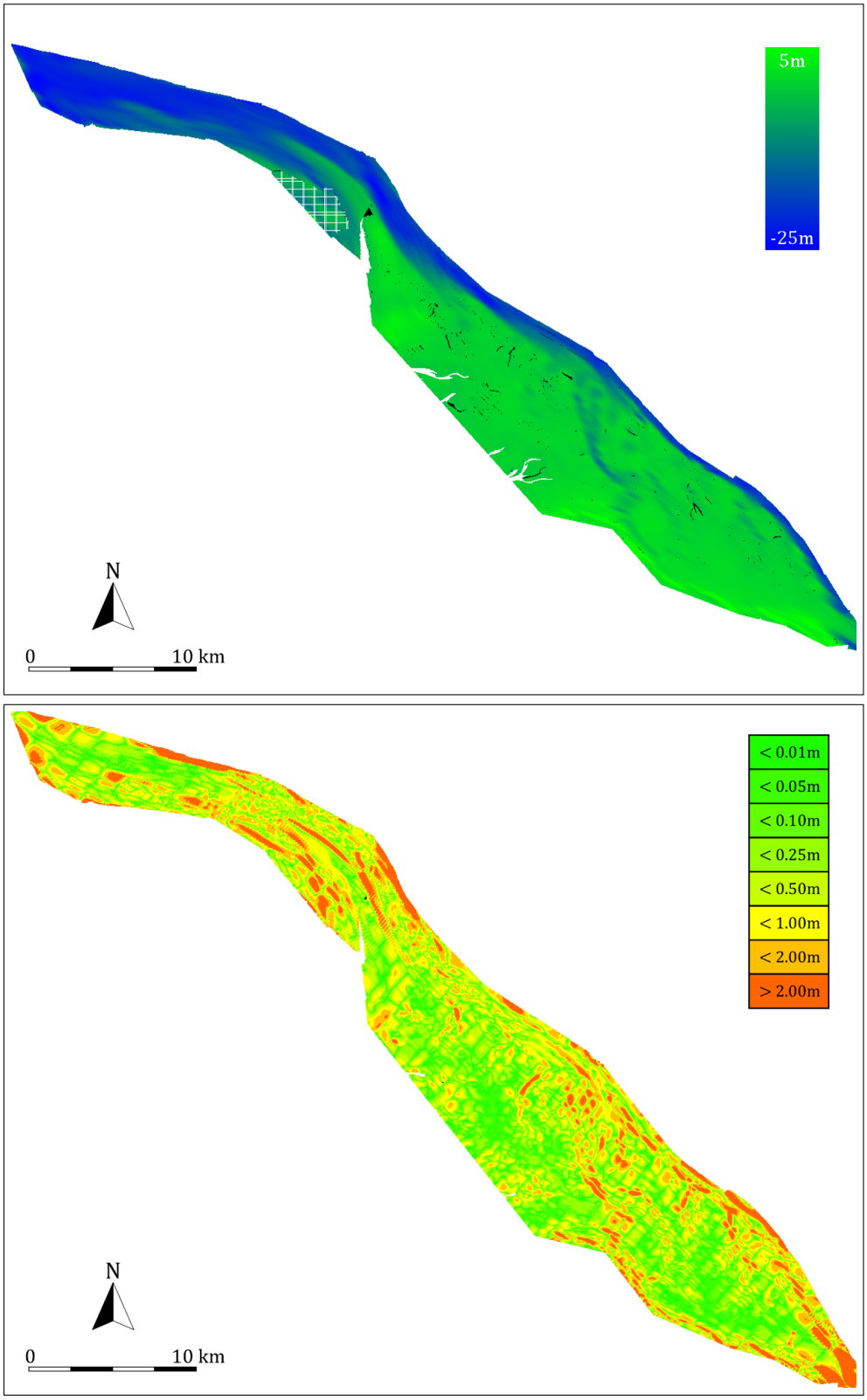


Abbildung 7.32: Kubische T-Spline-Fläche mit dem Dragging-Verfahren (5. Iteration)

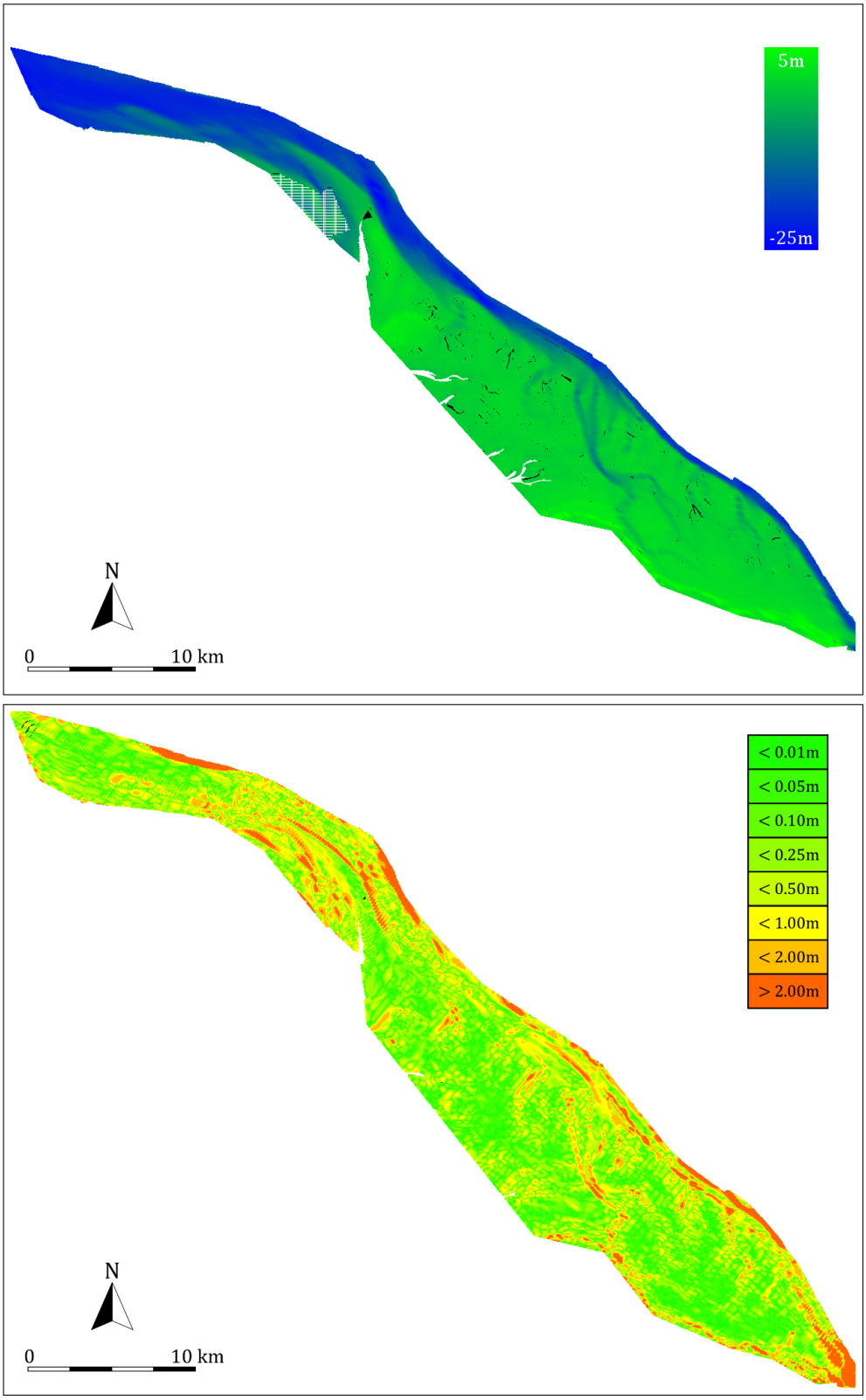


Abbildung 7.33: Kubische T-Spline-Fläche mit dem Dragging-Verfahren (6. Iteration)

8 Fazit

Die vorliegende Dissertation hat Ideen aus unterschiedlichen Fachgebieten zusammengeführt. Dieses Kapitel fasst die wesentlichen Erweiterungen der untersuchten Forschungsgebiete zusammen. Weitere Aspekte und offene Fragen folgen im Anschluss.

8.1 Zusammenfassung

Verschiedene Anwendungen des Ingenieurwesens beruhen auf Digitalen Geländemodellen. Im Kontext der bathymetrischen Geländemodelle motiviert diese Arbeit die Modellierung mit T-Spline-Flächen. Diese ermöglichen es konzeptbedingt, lokale Geländeeigenschaften innerhalb großräumiger Bathymetrien und die Beachtung der parametrischen Stetigkeit von T-Spline-Flächen zusammenzuführen.

Das Ziel war, eine vorhandene Bathymetrie mit hoher Datenmenge auf eine komprimierte Darstellung zu reduzieren. Ein mehrstufiger Prozess beschreibt den Vorgang, die Bathymetrie in ein T-Netz zu überführen. T-Netze sind hierbei die Kontrollpunktnetze für T-Spline-Flächen. Der Prozess unterscheidet zwischen einem direkten und einem indirekten Vorgehen. Die gleichfalls vorgestellte indirekte Herangehensweise umklammert den direkten Ablauf mit einer Vor- und einer Nachbearbeitung (Hin- und Rücktransformation) beruhend auf Algorithmen der Bildverarbeitung wie Warping und Dilatation.

Den Kern der Arbeit bildet ein Approximationsschema für die Flächenrückführung, angepasst auf verallgemeinerte Nicht-Standard-T-Spline-Flächen. Dieser integrierte Arbeitsschritt führt stufenweise eine Nicht-Standard-T-Spline-Fläche an die gegebene Bathymetrie heran, bis eine gewünschte Qualität erreicht ist. Dies geschieht wahlweise über die lineare Optimierung mit kleinsten Quadraten oder über ein modifiziertes Dragging-Verfahren für T-Spline-Flächen.

Vielfältige Operationen innerhalb der Arbeit weisen einen hohen Anteil an Parallelität auf. Anwender können dank GPGPU-Schnittstellen auf massiv-parallele Vektorrechner wie Grafikprozessoren zugreifen. Eigenständige OpenCL-Implementierungen ergänzen den Ablauf im Zusammenhang mit der T-Spline-Modellierung, der Flächenrückführung und dem Dragging-Verfahren.

Eine Demonstration am realen Anwendungsfall zeigt die generelle Machbarkeit des Verfahren auf. Hierbei werden praktische Grenzen ausgetestet und qualitative Tendenzen nachgewiesen, so dass erzeugte Resultate mit erwünschter Güte realistisch sind.

Insgesamt umschließt diese Arbeit vielfältige Methoden und Anwendungsgebiete, deren Kombinationen bislang kaum oder nicht untersucht sind. Der Wunsch liegt darin, Anwender an die zusätzlichen Möglichkeiten durch GPGPU-Beschleunigung heranzuführen und diese in weiteren Forschungsgebieten einzubinden.

8.2 Ausblick und Ergänzungen

Die bisherige Modellierung erfasst keine anthropogenen Elemente wie Bauwerke und Küstenschutzmaßnahmen. Es ist schwierig, eine exakte Deichlinie oder den Verlauf

einer Buhne zu erfassen. Zorndt et al. [100] erreichen dies durch semi-automatisierte Neuvernetzung von Dreiecknetzen. T-Spline-Flächen erlauben die direkte Modellierung von C^0-stetigen Unebenheiten in einer Geometrie. Weitergehende Arbeiten können derartige Modelle aus einer „zweiten Ebene“ sinnvoll integrieren, um somit ein konsistentes Gesamtmodell zu erstellen.

Weitreichende Studien über die Parametrisierung des vorgestellten Verfahrens waren kein Teil dieser Forschungsarbeit. Zukünftig können Hypothesen über weitere praktische Modellgrenzen erstellt und geprüft werden. Gibt es beispielsweise Verhältnisse untereinander in den Eingangsgrößen (Netzgröße, Messpunktdichte, Varianz der Tiefenwerte, etc.), in denen Regelmäßigkeiten, bezogen auf das qualitative und quantitative Ergebnis der Reduktion bathymetrischer Flächen, aufspürbar sind ? Inwieweit sind diese vom betrachteten Szenario abhängig oder sogar allgemeingültig ? Verbesserungen des Verfahrens können darüber hinaus auf gefensterte Abläufe zurückgreifen, d. h. die zu reduzierenden Gebiete der Bathymetrie werden durch einen verschiebbaren Ausschnitt betrachtet und die jeweiligen Verfahren laufen somit auf verringerten Eingabegrößen ab.

Wesentliches Potential steckt in der indirekten Reduktion. In der vorgestellten Arbeit wird das gesamte Einzugsgebiet der vermessenen Bathymetrie betrachtet. Anstelle dessen können die relevanten Gebiete (z. B. Fahrrinnen) gezielt analysiert und für die Flächenrückführung modelliert werden. Weitergehende Untersuchungen können die Auswirkungen der gezielten Auswahl von Gebieten für die Bestimmung der globalen Orientierung betrachten. Ferner erfolgt die Berechnung der Warping-Transformation somit ausschließlich auf diesen charakteristischen Gebieten.

Die Abläufe der globalen und lokalen Transformation greifen auf die Warping-Heuristik zur Berechnung von Verschiebungsvektoren zurück. Alternative Strategien für die räumfüllende Verteilung einer Punktmenge ergeben sich z. B. aus der Theorie der *Snake*-Spline-Kurven (energetischer Ausgleich zwischen maximaler Ausdehnung und minimierter Krümmung [53]) oder aus approximativen Algorithmen wie dem ICP-Schema (*iterative closest point*) [70].

Ursprünglich führen Sederberg et al. [77] bereits in ihrer Originalarbeit sogenannte T-NURCC-Flächen (*Non-Uniform Rational Catmull-Clark surface*) als Unterteilungsflächen angelehnt an Catmull und Clark [17] mit sternförmigen Kreuzungspunkten ein. Ein Vorteil der T-NURCC-Flächen besteht in der frei wählbaren Zerlegung von Zellen unter Beibehaltung der Spline-Eigenschaften. Weitergehende Untersuchen, gerade in Hinblick auf die Praxistauglichkeit, stellen die Autoren allerdings nicht an. Es wäre zu untersuchen, ob küstennahe Bauten mit T-NURCC-Flächen effektiver modelliert werden (z. B. entlang der o. g. Deichkanten).

Das OpenCL-Modell sieht keine Synchronisation an globalen Barrieren zwischen sämtlichen verfügbaren Rechenkernen aller Work Groups vor. Vergleichbares gilt für die CUDA-Bibliothek. Es ist nicht auszuschließen, dass spätere Versionen der Schnittstelle dieses Manko beheben. In Hinblick auf die globale Synchronisation des Dragging-Verfahrens (Abschnitt 6.3.3) wäre es von immensem Vorteil, wenn diese Device-intern und somit unter Ausschluss der zusätzlichen Datentransfers stattfindet. Dies erlaubt eine vollständige Implementierung des Dragging-Algorithmus auf GPGPU-Basis. Sollte dieses Problem zukünftig aufgehoben werden, wäre sogar ein durchgehender Prozess der Flächenreduktion vollständig auf GPGPU-Basis denkbar.

Literaturverzeichnis

[1] Abramowski, Stephan und Heinrich Müller: *Geometrisches Modellieren*. BI-Wiss.Verl., Mannheim; Wien; Zürich, 1991.

[2] Achakeev, Daniar, Marc Seidemann, Markus Schmidt und Bernhard Seeger: *Sort-based Parallel Loading of R-trees*. In: *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, BigSpatial '12, Seiten 62–70, New York, NY, USA, 2012. ACM.

[3] Aizeng, Wang und Zhao Gang: *The Analysis of T-spline Fairness*. Computer-Aided Design and Applications, 10(5):817–825, jan 2013.

[4] Alpert, N M, J F Bradshaw, D N Kennedy und J A Correia: *The Principal Axes Transformation - A method for Image Registration*. Journal of Nuclear Medicine, 31:1717–1722, 1990.

[5] Asche, Christian: *Wavelet-basierte Kompression von bathymetrischen Geländedaten*. In: *Forum Bauinformatik*, 2009.

[6] Asche, Christian und Volker Berkhahn: *Efficient data structures for T-spline modelling*. In: Borrmann, André, Philipp Geyer, Yaqub Rafiq und Pieter de Wilde (Herausgeber): *International Workshop: Intelligent Computing in Engineering*. Technische Universität München, 2012.

[7] Asche, Christian und Volker Berkhahn: *T-spline surface design in engineering*. In: Telichenko, Valery, Andrey Volkov und Irinia Bilchuk (Herausgeber): *14th International Conference on Computing in Civil and Building Engineering*. The Publishing House, 2012.

[8] Asche, Christian und Axel Rauschenberger: *Flächenmodellierung mit T-Splines im Ingenieurwesen*. In: Tobin, Ena und Michal Otreba (Herausgeber): *Forum Bauinformatik*, Seite 8, Cork, 2011. University College Cork.

[9] Beierle, Christoph und Gabriele Kern-Isberner: *Methoden wissensbasierter Systeme*. Springer Fachmedien Wiesbaden, 2014.

[10] Berkhahn, Volker: *Geometrisches Modellieren in der Bauinformatik*. Habilitationsschrift, Leibniz Universität Hannover, 2005.

[11] Bertram, Martin, Xavier Tricoche und Hans Hagen: *Adaptive smooth scattered-data approximation for large-scale terrain visualization*. In: *Proceedings of the Symposium on Data Visualisation*, Seiten 177–184. Eurographics Association, 2003.

[12] Beyer, G.: *Wavelettransformationen hybrider Geländemodelle*. Doktorarbeit, Technische Universität Dresden, 2005.

[13] Bowyer, A.: *Computing Dirichlet tessellations*. The Computer Journal, 24(2):162–166, 1981.

[14] Burkard, Rainer E. und Uwe T. Zimmermann: *Einführung in die Mathematische Optimierung.* Springer Berlin Heidelberg, 2008.

[15] Cardon, David L.: *T-Spline Simplification.* Doktorarbeit, Brigham Young University, 2007.

[16] Carlson, Nils und M. Gulliksson: *Surface Fitting with NURBS: A Gauss Newton with Trust Region Approach.* In: *Proceedings of the 13th WSEAS International Conference on Applied Mathematics*, MATH'08, Seiten 169–174, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[17] Catmull, Edwin und James Clark: *Recursively generated B-spline surfaces on arbitrary topological meshes.* Computer-aided design, 10(6):350—-355, 1978.

[18] Chivate, Pramod N. und Andrei G. Jablokow: *Review of surface representations and fitting for reverse engineering.* Computer Integrated Manufacturing Systems, 8(3):193–204, aug 1995.

[19] Corporation, Nvidia: *NVidia CUDA*, 2014.

[20] Cottrell, J. Austin, Thomas J. R. Hughes und Yuri Bazilevs: *Isogeometric analysis: toward integration of CAD and FEA.* John Wiley & Sons, Ltd, 2009.

[21] De Floriani, Leila, David Greenfieldboyce und Annie Hui: *A Data Structure for Non-manifold Simplicial d-complexes.* In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, Seiten 83–92, New York, NY, USA, 2004. ACM.

[22] Deng, Jiansong, Falai Chen, Xin Li, Changqi Hu, Weihua Tong, Zhouwang Yang und Yuyu Feng: *Polynomial splines over hierarchical T-meshes.* Graphical Models, 70(4):76–86, jul 2008.

[23] Eck, Matthias und Jan Hadenfeld: *Knot Removal for B-Spline Curves.* Computer Aided Geometric Design, 12(3):259–282, 1995.

[24] Eck, Matthias und Hugues Hoppe: *Automatic reconstruction of B-spline surfaces of arbitrary topological type.* In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, Seiten 325–334, New York, New York, USA, 1996. ACM Press.

[25] Fabri, Andreas und Sylvain Pion: *CGAL - Computational Geometry Algorithms Library*, 2009.

[26] Farin, Gerald: *NURBS for Curve & Surface Design: From Projective Geometry to Practical Use.* Ak Peters Series. Taylor & Francis, 1999.

[27] Farin, Gerald: *Curves and Surfaces for CAGD: A Practical Guide.* Computer graphics and geometric modeling. Morgan Kaufmann, 2002.

[28] Fogel, E., D. Halperin und R. Wein: *CGAL Arrangements and their Applications.* Springer Heidelberg Dordrecht London New York, 2012.

[29] Franke, Richard und Gregory M. Nielson: *Scattered Data Interpolation and Applications: A Tutorial and Survey.* In: Hagen, Hans und Dieter Roller (Herausgeber): *Geometric Modeling*, Seiten 131–160. Springer Berlin Heidelberg, 1991.

[30] Gaster, Benedict, Lee Howes, David R. Kaeli, Perhaad Mistry und Dana Schaa: *Heterogeneous Computing with OpenCL.* Elsevier Science & Technology, 2012.

[31] Glasbey, C. A. und K. V. Mardia: *A review of image-warping methods.* Journal of Applied Statistics, 25(2):155–171, 1998.

[32] Goshtasby, A.: *Piecewise cubic mapping functions for image registration.* Pattern Recognition, 20:525, 1987.

[33] Group, Khronos: *Khronos Group*, 2014.

[34] Gu, Xianfeng, Ying He und Hong Qin: *Manifold splines.* Graphical Models, 68(3):237–254, may 2006.

[35] Guttman, A.: *R-trees: a dynamic index structure for spatial searching.* In: *SIGMOD '84 Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, Seiten 47–57, 1984.

[36] He, Ying, Kexiang Wang, Hongyu Wang, Xianfeng Gu und Hong Qin: *Manifold T-Spline.* In: *Proceedings of Geometric Modeling and Processing*, Band 68, Seiten 409–422, may 2006.

[37] Heidrich, Wolfgang, Richard Bartels und George Labahn: *Fitting uncertain data with NURBS.* Proceedings 3rd Int. Conf. on Curves and Surfaces in Geometric Design, Seiten 1–8, 1996.

[38] Hoschek, Josef und Ulrich Dietz: *Smooth B-Spline Surface Approximation to Scattered Data.* In: Hoscheck, Josef und Werner Dankwort (Herausgeber): *Reverse Engineering*, Seiten 143–152. Vieweg+Teubner Verlag, 1996.

[39] Hoschek, Josef und Dieter Lasser: *Grundlagen der geometrischen Datenverarbeitung.* Vieweg+Teubner Verlag, 1992.

[40] Hughes, Thomas J. R., J. Austin Cottrell und Yuri Bazilevs: *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement.* Computer Methods in Applied Mechanics and Engineering, 194(39-41):4135–4195, oct 2005.

[41] Hwu, Wen-Mei: *GPU Computing Gems Emerald Edition.* Morgan Kaufmann, 2011.

[42] Ipson, Heather: *T-Spline Merging.* Doktorarbeit, Brigham Young University, 2005.

[43] Kaapke, Kai: *Entwurfsorientierte Flächenrückführung.* Verlag Dr. Hut, 2009.

[44] Kenny, A. J., I. Cato, M. Desprez, G. Fader, R. T. E. Schu und J. Side: *An overview of seabed-mapping technologies in the context of marine habitat classification.* ICES Journal of Marine Science, 3139(03):411–418, 2003.

[45] Klein, Gregory J.: *Forward Deformation of PET Volumes Using Non-uniform Elastic Material Constraints.* In: Kuba, Attila, Martin Šáamal und Andrew Todd-Pokropek (Herausgeber): *Information Processing in Medical Imaging*, Seiten 358–363. Springer Berlin Heidelberg, 1999.

[46] Kobbelt, Leif und Mario Botsch: *A Survey of Point-based Techniques in Computer Graphics.* Comput. Graph., 28(6):801–814, 2004.

[47] Krishnamurthy, Adarsh, Rahul Khardekar und Sara McMains: *Direct evaluation of NURBS curves and surfaces on the GPU.* Proceedings of the 2007 ACM symposium on Solid and physical modeling, 1(212):329–334, 2007.

[48] Krishnamurthy, Adarsh, Rahul Khardekar, Sara McMains, Kirk Haller und Gershon Elber: *Performing efficient NURBS modeling operations on the GPU.* IEEE transactions on visualization and computer graphics, 15(4):530–43, 2008.

[49] Krishnamurthy, Adarsh, S. McMains und K. Halle: *Accelerating geometric queries using the GPU.* Proceedings of the 2009 ACM Symposium on Solid and Physical Modeling, Seiten 199–210, 2009.

[50] Krishnamurthy, Adarsh und Sara McMains: *Accurate moment computation using the GPU.* Proceedings of the 14th ACM Symposium on Solid and Physical Modeling - SPM '10, Seite 81, 2010.

[51] Kshemkalyani, Ajay D. und Mukesh Singhal: *Distributed computing: principles, algorithms, and systems.* Cambridge University Press, 2008.

[52] Lai, Yu-Kun, Shi-Min Hu und Helmut Pottmann: *Surface fitting based on a feature sensitive parametrization.* Computer-Aided Design, 38(7):800–807, jul 2006.

[53] Lee, Seung-Yong, Kyung-Yong Chwa und Sung Yong Shin: *Image metamorphosis using snakes and free-form deformations.* In: *SIGGRAPH '95 Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, Seiten 439–448. ACM, 1995.

[54] Li, Wan-Chiu, Nicolas Ray und Bruno Lévy: *Automatic and Interactive Mesh to T-Spline Conversion.* In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, Seiten 191–200, Cagliari, Sardinia, Italy, 2006. Eurographics Association.

[55] Li, Z., C. Zhu und C. Gold: *Digital Terrain Modeling: Principles and Methodology.* Taylor & Francis, 2004.

[56] Lin, Hongwei, Ye Cai und Shuming Gao: *Extended T-mesh and Data Structure for the Easy Computation of T-spline.* Journal of Information & Computational Science, 3(March):583–593, 2012.

[57] Lin, Hongwei und Zhiyu Zhang: *An efficient method for fitting large data sets using T-splines.* SIAM Journal on Scientific Computing, 35(6):3052–3068, 2013.

[58] Lodha, S.K. und R. Franke: *Scattered Data Techniques for Surfaces.* In: *Scientific Visualization Conference*, 1997.

[59] Lyche, Tom und Knut Morken: *Knot removal for parametric B-spline curves and surfaces.* Computer Aided Geometric Design, 4(3):217–230, nov 1987.

[60] Ma, Weiyin und Jean-Pierre Kruth: *Mathematical Modeling of Free-form Curves and Surfaces from Discrete Points with NURBS.* In: *Proceedings of the International Conference on Curves and Surfaces in Geometric Design*, Seiten 319–326, Natick, MA, USA, 1994. A. K. Peters, Ltd.

[61] Ma, Weiyin und Jean-Pierre Kruth: *NURBS curve and surface fitting for reverse engineering.* The International Journal of Advanced Manufacturing Technology, 14:918–927, 1998.

[62] Milbradt, Peter, Cathrin Dorow und Christian Asche: *Identifikation morphologischer Tendenzen und Geschwindigkeiten im Küstennahbereich: Abschlussbericht; Berichtszeitraum: 01.10.2006 - 31.09.2009.* Leibniz Universität Hannover, 2009.

[63] Moore, D. und Joe Warren: *Approximation of dense scattered data using algebraic surfaces.* In: *System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on*, Band i, Seiten 681–690 vol.1, jan 1991.

[64] Munshi, Aaftab, Benedict Gaster, Timothy G. Mattson, James Fung und Dan Ginsburg: *OpenCL Programming Guide.* Addison-Wesley Professional, 1 Auflage, 2011.

[65] Papageorgiou, Markos, Marion Leibold und Martin Buss: *Optimierung - Statische, Dynamische, Stochastische Verfahren für die Anwendung.* Springer Berlin Heidelberg, 2012.

[66] Piegl, Les A. und Wayne Tiller: *The NURBS book.* Monographs in visual communication. Springer, 1995.

[67] Pottmann, Helmut, Tibor Steiner, Michael Hofer, Christoph Haider und Allan Hanbury: *The isophotic metric and its application to feature sensitive morphology on surfaces.* In: Pajdla, Tomás und Jiří Matas (Herausgeber): *Computer Vision - ECCV 2004*, Seiten 560–572. Springer Berlin Heidelberg, 2004.

[68] Prasad, Sushil K., Shashi Shekhar, Michael McDermott, Xun Zhou, Michael Evans und Satish Puri: *GPGPU-accelerated Interesting Interval Discovery and Other Computations on GeoSpatial Datasets - A Summary of Results.* In: *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, BigSpatial '13, Seiten 65–72, New York, NY, USA, 2013. ACM.

[69] Ruprecht, Detlef und Heinrich Müller: *Image warping with scattered data interpolation methods.* IEEE Computer Graphics and Applications, 15(2):37–43, 1995.

[70] Rusinkiewicz, S. und M. Levoy: *Efficient variants of the ICP algorithm.* Proceedings Third International Conference on 3-D Digital Imaging and Modeling, Seiten 145–152, 2001.

[71] Savu, Adrian und Caius Didulescu: *3d modelling using laser scanning technique.* Journal of Applied Engineering Sciences, 1(1):119–124, 2013.

[72] Scarpino, Matthew: *OpenCL in Action: How to Accelerate Graphics and Computations.* Manning Publications, nov 2011.

[73] Scott, Michael A., Xin Li, Thomas W. Sederberg und Thomas J. R. Hughes: *Local refinement of analysis-suitable T-splines.* Computer Methods in Applied Mechanics and Engineering, 213-216:206–222, mar 2012.

[74] Sederberg, Matthew T. und Thomas W. Sederberg: *T-Splines: A Technology for Marine Design with Minimal Control Points.* The second Chesapeake power boat symposium proceedings, 26:250–255, 2010.

[75] Sederberg, Thomas W. und David L. Cardon: *T-spline simplification and local refinement.* ACM Transactions on Graphics, 54(2), 2004.

[76] Sederberg, Thomas W., Gordon Thomas Finnigan, Xin Li, Hongwei Lin und Heather Ipson: *Watertight trimmed NURBS.* ACM Transactions on Graphics, 27(3):1, aug 2008.

[77] Sederberg, Thomas W., Jianmin Zheng, Almaz Bakenov und Ahmad Nasri: *T-splines and T-NURCCs.* ACM Trans. Graph, 22(3):477, 2003.

[78] Serra, Jean: *Image analysis and mathematical morphology.* Cytometry, 4, 1982.

[79] Taubman, David und Michael Marcellin: *JPEG2000 Image Compression Fundamentals, Standards and Practice.* Springer US, 2002.

[80] Tay, Raymond: *OpenCL Parallel Programming Development Cookbook.* Packt Publishing, 2013.

[81] Tiller, Wayne: *Knot-removal algorithms for NURBS curves and surfaces.* Computer Aided Design, 24(8):445–453, 1992.

[82] Wang, Hongyu, Ying He, Xin Li, Xianfeng Gu und Hong Qin: *Geometry-aware domain decomposition for T-spline-based manifold modeling.* Computers & Graphics, 33(3):359–368, 2009.

[83] Wang, Yimin und Jianmin Zheng: *Control point removal for T-Spline Surfaces.* In: Kim, Myung-Soo und Kenji Shimada (Herausgeber): *Geometric Modeling and Processing - GMP 2006*, Seiten 385–396. Springer Berlin Heidelberg, 2006.

[84] Wang, Yimin und Jianmin Zheng: *Curvature-guided adaptive T-spline surface fitting.* Computer-Aided Design, 45(8-9):1095–1107, aug 2013.

[85] Weiss, V., L. Andor, G. Renner und T. Várady: *Advanced Surface Fitting Techniques.* Comput. Aided Geom. Des., 19(1):19–42, 2002.

[86] Wolberg, George: *Digital image warping.* IEEE computer society press Los Alamitos, CA, 1990.

[87] Wolberg, George: *Image morphing: a survey.* The Visual Computer, 14(8-9):360–372, dec 1998.

[88] Wriggers, Peter, Udo Nackenhorst, Sascha Beuermann, Holger Spiess und Stefan Löhnert: *Technische Mechanik kompakt*. Vieweg+Teubner Verlag, 2006.

[89] Yang, Huaiping, Matthias Fuchs, Bert Jüttler und Otmar Scherzer: *Evolution of T-Spline Level Sets with Distance Field Constraints for Geometry Reconstruction and Image Segmentation*. In: *Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006*, SMI '06, Washington, DC, USA, 2006. IEEE Computer Society.

[90] Yang, Huaiping und Bert Jüttler: *Evolution of T-spline Level Sets for Meshing Non-uniformly Sampled and Incomplete Data*. The Visual Computer, 24(6):435–448, 2008.

[91] You, Simin, Jianting Zhang und Le Gruenwald: *Parallel Spatial Query Processing on GPUs Using R-trees*. In: *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, BigSpatial '13, Seiten 23–31, New York, NY, USA, 2013. ACM.

[92] Zhang, Jianting: *Towards Personal High-performance Geospatial Computing (HPC-G): Perspectives and a Case Study*. In: *Proceedings of the ACM SIGSPATIAL International Workshop on High Performance and Distributed Geographic Information Systems*, HPDGIS '10, Seiten 3–10, New York, NY, USA, 2010. ACM.

[93] Zhang, Jianting: *Speeding Up Large-scale Geospatial Polygon Rasterization on GPGPUs*. In: *Proceedings of the ACM SIGSPATIAL Second International Workshop on High Performance and Distributed Geographic Information Systems*, HPDGIS '11, Seiten 10–17, New York, NY, USA, 2011. ACM.

[94] Zhang, Jianting und Simin You: *High-performance quadtree constructions on large-scale geospatial rasters using GPGPU parallel primitives*. International Journal of Geographical Information Science, 27(11):2207–2226, 2013.

[95] Zhang, Jianting, Simin You und Le Gruenwald: *Indexing Large-scale Raster Geospatial Data Using Massively Parallel GPGPU Computing*. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, Seiten 450–453, New York, NY, USA, 2010. ACM.

[96] Zhang, Jianting, Simin You und Le Gruenwald: *Parallel Quadtree Coding of Large-scale Raster Geospatial Data on GPGPUs*. In: *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, Seiten 457–460, New York, NY, USA, 2011. ACM.

[97] Zhao, Xiang Jun, Mei Lu und Hong Xin Zhang: *Patch Reconstruction with T-Splines Iterated Fitting for Mesh Surface*. Applied Mechanics and Materials, 88-89:491–496, aug 2011.

[98] Zheng, Jianmin, Yimin Wang und H. S. Seah: *Adaptive T-spline surface fitting to z-map models*. In: *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, Nummer 2, Seiten 405–411. ACM, 2005.

[99] Zorin, Denis: *Constructing Curvature-continuous Surfaces by Blending.* In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, Seiten 31–40, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[100] Zorndt, Anna, Torsten Schlurmann und Iris Grabemann: *The influence of extreme events on hydrodynamics and salinities in the Weser estuary in the context of climate impact research.* Coastal Engineering Proceedings, 1(33):1–12, 2012.

[101] Zorndt, Anna, Andreas Wurpts und Torsten Schlurmann: *Aufbau und Kalibrierung eines 3d-hydromechanischen Ästuarmodells zur Abbildung der Salzintrusion in Tidebeeinflussten Gewässern unter Berücksichtigung von Klimaszenarien.* In: *Tagungsband des 8. FZK-Kolloquiums Maritimer Wasserbau und Küsteningenieurwesen*, 2011.

Abbildungsverzeichnis

Tabellenverzeichnis

Index

Wissenschaftlicher Werdegang

Persönliches

Name:	Christian Asche
geboren:	19. Juni 1978 in Bochum
Nationalität:	deutsch
Familienstand:	verheiratet

Wissenschaftlicher Werdegang

1984 – 1988	Grundschule Claudiusstraße in Herne
1988 – 1997	Gymnasium Wanne in Herne Erwerb der allgemeinen Hochschulreife
1997 – 2008	Studium an der Technischen Universität Dortmund Abschluss „Diplom Kern-Informatik“ an der „Fakultät für Informatik“ Nebenfach „Architektur“ an der „Fakultät Architektur und Bauingenieurwesen“
15.08.2005 – 31.03.2006, 01.09.2006 – 28.02.2007	Wissenschaftliche Hilfskraft am Lehrstuhl „Grafische Systeme“ an der Technischen Universität Dortmund
01.10.2007 – 31.03.2008	Wissenschaftliche Hilfskraft am Lehrstuhl „Logik in der Informatik“ an der Technischen Universität Dortmund
01.10.2008 - 30.09.2014	Wissenschaftlicher Mitarbeiter am „Institut für Bauinformatik“ der Gottfried Wilhelm Leibniz Universität Hannover