Eine geometrieunabhängige Strategie zur Filterentwicklung in der 3D-Tomographie und ihre Implementierung mittels objektorientierter Methoden

Dissertation

zur Erlangung des Grades

des Doktors der Naturwissenschaften

der Naturwissenschaftlich-Technischen Fakultät I

der Universität des Saarlandes

von

Roman Müller

Saarbrücken

2002

Tag des Kollquiums:	12. Juli 2002
Dekan:	Univ.-Prof. Dr. P. Slusallek
Berichterstatter:	Univ.-Prof. Dr. A. K. Louis
	Univ.-Prof. Dr. J. Weickert

En giro torte sol ciclos et rotor igne.

Inhaltsverzeichnis

Einleitung **1**

1 Eine neue Rekonstruktionsstrategie **9**

1.1 Grundlagen der 3D-Computer-Tomographie 9

1.1.1 Problemstellung und Verfahrensbeschreibung 9

1.1.2 Das mathematische Modell 14

1.1.3 Die Approximative Inverse 19

1.2 Inversionsformeln und Rekonstruktion 23

1.2.1 Anwendung der AI auf die Computertomographie: Inversion und Filterberechnung 23

1.2.1.1 RADON-Transformation 23

1.2.1.2 RÖNTGEN-Transformation 26

1.2.1.3 Divergente RÖNTGEN-Transformation 27

1.3 Das GUF-Verfahren 30

1.3.1 Motivation 30

1.3.2 Verfahrensbeschreibung 33

1.3.3 GUF-Verfahren und Regularisierung 38

1.3.4 Voruntersuchung: der 2D-Fall 45

1.3.5 Basisfunktionen im 2D-Fall 46

1.3.5.1 Charakteristische Funktion des Kreises 46

1.3.5.2 Charakteristische Funktion des Quadrates 46

1.3.5.3 Radiale Gauss-Funktion in 2D 48
1.3.5.4 Pyramidale Pixel-Basis 48
1.3.6 Basisfunktionen im 3D-Fall 51
1.3.6.1 Charakteristische Funktion der Kugel 51
1.3.6.2 Charakteristische Funktion des Würfels 52
1.3.6.3 Radiale Gauss-Basis in 3D 52
1.3.6.4 Pyramidale Voxel-Basis 53
1.4 Rekonstruktionen 59
1.4.1 Voruntersuchung: Der 2D-Fall 59
1.4.2 Der 3D-Fall 60
1.4.2.1 Approximation an vollständige Daten 64
1.4.2.2 Tuy-Kirillov-Geometrien 66
1.4.2.3 Andere Geometrien 75

2 Implementierung:
Das TORNE Framework **87**

2.1 Einführung 87
2.2 OOP und Muster 90
2.2.1 Programmierparadigmen 90
2.2.2 OO-Modellierung und Design 95
2.2.3 Entwurfsmuster 97
2.3 Design 99
2.3.1 Die Muster in TORNE 99
2.3.2 Weitergehende Designfragen 106
2.3.3 Beispielnetzwerk und Ablaufkontrolle 109
2.4 Rekonstruktionsalgorithmen 113
2.4.1 Approximative Inverse 113
2.4.2 Tomosynthese 116
2.4.3 GUF-Verfahren 116

3 Objekt- und Musterkatalog **123**

3.1 Übersicht 123

3.2 Katalog 128

3.2.1 Containerklassen 128

3.2.2 Prozessklassen 132

A Das verwendete Phantom **155**

Bibliographie **157**

Abbildungsverzeichnis

1.1 Datengewinnung 10
1.2 Radialbasis 47
1.3 Radontransformation der Radialbasis 47
1.4 Quadratbasis 48
1.5 Radontransformation der Quadratbasis 48
1.6 Gaussbasis 49
1.7 Radontransformation der Gaussbasis 49
1.8 Pyramidalbasis 50
1.9 Radontransformation der Pyramidalbasis 50
1.10 Rekonstruktion mit char. Funktion des Kreises 59
1.11 Rekonstruktion mit radialer Gauss-Basis 60
1.12 Rekonstruktion mit char. Funktion des Quadrates 60
1.13 Rekonstruktion mit pyramidaler Pixel-Basis 61
1.14 Das verwendete Phantom 63
1.15 Monte-Carlo-Geometrie 64
1.16 Rekonstruktion für die Monte-Carlo-Geometrie 65
1.17 Tuy-Kirillov-Charakterisierung 66
1.18 TK-Kreis-Geometrie 67
1.19 Rekonstruktion für die Tuy–Kirillov–Kreis–Geometrie 68
1.20 Sinusoid-Geometrie 70
1.21 Rekonstruktion für die Sinusoid–Geometrie 70

1.22 Sphärische-Spiral-Geometrie 71
1.23 Rekonstruktion für die sphärische Spiral-Geometrie 72
1.24 Spiral-Geometrie 73
1.25 Rekonstruktion für die Spiral-Geometrie mit 4 Umläufen 74
1.26 Cone-Beam-Geometrie 76
1.27 Rekonstruktion für die Cone–Beam–Geometrie (synth. Daten) 77
1.28 Rekonstruktion realer Daten: Basisgrösse 500 78
1.29 Rekonstruktion realer Daten: Basisgrösse 500, alt. Darstellung 78
1.30 Rekonstruktion realer Daten: Basisgrösse 1000 79
1.31 Rekonstruktion realer Daten: Basisgrösse 1000, alt. Darstellung 79
1.32 Rekonstruktion mit gefilterter Rückprojektion und Approximativer Inverse 80
1.33 Rekonstruktion mit der Gauss-Basisfunktion 81
1.34 Rekonstruktion mit der radialen χ-Basisfunktion 81
1.35 Rekonstruktion mit der Würfel-χ-Basisfunktion 82
1.36 Rekonstruktion mit paralleler Geometrie und pyramidaler Basisfunktion . . 82
1.37 Tomosynthese-Geometrie 83
1.38 Rekonstruktion für die Tomosynthese-Geometrie 84
1.39 Filter für Conebeam, xy-Schnitt 85
1.40 Filter für Tomosynthese, xy-Schnitt 85
1.41 Filter für Conebeam, xz-Schnitt 86
1.42 Filter für Tomosynthese, xz-Schnitt 86
1.43 Filter für Conebeam, yz-Schnitt 86
1.44 Filter für Tomosynthese, yz-Schnitt 86

2.1 Grundstruktur von TORNE 101
2.2 CContainer 103
2.3 CProcess 104
2.4 Beispielprozessnetz 110
2.5 Update-Execute-Zyklus 112

2.6 AI Objektmodell 114
2.7 AI Dynamikmodell 114
2.8 AI Funktionalmodell 115
2.9 Tomosynthese Objektmodell 117
2.10 Tomosynthese Dynamikmodell 117
2.11 Tomosynthese Funktionalmodell 118
2.12 GUF Objektmodell 120
2.13 GUF Dynamikmodell 121
2.14 GUF Funktionalmodell 122

3.1 Container 123
3.2 Geometer und Controller 124
3.3 Filter und Mapper 125
3.4 Provider und Outlet 126

Einleitung

Die 3D-Computertomographie ist ein bildgebendes Verfahren, bei dem ein Untersuchungsobjekt mit einem Bündel aus Röntgenstrahlen bestrahlt wird. Durch die Rekonstruktion des dreidimensionalen Objektes unter Verwendung der zweidimensionalen Daten gewinnt man Informationen über dessen Inneres.
Das mathematische Modell führt zu einer Integralgleichung

$$\mathcal{D}f = g,$$

deren Inversion in die Klasse der schlechtgestellten Probleme fällt. Kleine Fehler in den Daten führen zu grossen Fehlern im Rekonstruktionergebnis. Daher wird eine Regularisierung angewendet.
Praktisch gesehen liegen niemals vollständige Daten vor. Die Daten werden immer an diskreten Quellpositionen im $\mathbb{R}^3$ gewonnen, was als Aufnahmegeometrie bezeichnet wird. Analytisch betrachtet bildet diese einen Weg Γ im Raum, real eine diskrete Punktmenge. Eine exakte Inversion ist ausser bei vollständigen Daten dann möglich, wenn die Aufnahmegeometrie die Tuy–Kirillov–Bedingung erfüllt. Sie spielt somit eine entscheidende Rolle für die Rekonstruktion.
Betrachtet man die Rekonstruktionsverfahren, so fallen die hauptsächlich verwendeten in drei Kategorien: direkte Fourier-Methoden, Filterung nach Rückprojektion und gefilterte Rückprojektion.
Die direkten Fourier-Methoden invertieren obige Integralgleichung auf eine mathematisch elegante Art und Weise, indem sie mit Hilfe des Fourier-Slice-Theorem aus den Daten direkt

Informationen über das Spektrum der gesuchten Funktion berechnen. Die Aufnahmegeometrie spiegelt sich in einem mehr oder minder dichten Spektrum wider. Die Funktion erhält man dann unter Verwendung einer inversen Fourier-Transformation. Das grundlegende Problem dieser Methode liegt hier darin, dass das Fourier-Slice-Theorem eine parallele Strahlanordnung statt einem kegelförmigen Strahlenbündel voraussetzt, was praktisch nicht erfüllt ist. Weitere Probleme stellen die unvermeidlichen Messfehler, die Interpolation des Funktionsspektrums und dessen Repräsentation auf einem Polargitter dar. Daher hat sich diese Methode in der Praxis nicht durchgesetzt.

Bei der Methode der Filterung nach Rückprojektion ist, wie der Name schon sagt, der erste Schritt der Inversion die Rückprojektion, d.h. die Abbildung der zweidimensionalen Daten in den dreidimensionalen Raum durch konstante Fortsetzung entlang der Strahlen. Die Filterung geschieht durch nachträgliche Differentiation im Orts- oder Frequenzraum, wobei letzteres aus Effizienzgründen zu bevorzugen ist. Die Geometrie geht hier wieder über die Rückprojektion in das Spektrum der rekonstruierten Funktion ein. Hier ergeben sich Probleme bei der numerischen Differentiation: die Daten sind in der Praxis verrauscht und müssen zusätzlich regularisiert werden; ausserdem geht der Durchschnitt der Rekonstruktion verloren. Der Filterungsschritt erfolgt weiterhin ungeachtet der Aufnahmegeometrie immer gleich, so dass auch Fehler im Spektrum gewichtet werden. Diese Methode hat sich praktisch ebenfalls nicht durchgesetzt.

Die meistverwendete Rekonstruktionsmethode ist die gefilterte Rückprojektion. Hier werden die Daten vor der Rückprojektion mit Hilfe eines Faltungsfilters vorbehandelt, was Sprünge verstärkt, die durch den Vorgang der Rückprojektion wieder geglättet werden. Die Aufnahmegeometrie geht über zwei Funktionen in die Filterberechnung ein, von denen eine die Richtung der Aufnahmekurve modelliert und die andere die Anzahl der Schnittpunkte einer bestimmten Ebene durch einen Rekonstruktionspunkt mit der Aufnahmekurve. Hier geht im Gegensatz zu den vorherigen Rekonstruktionsverfahren die Aufnahmegeometrie explizit, nicht nur implizit, in die Kernberechnung ein. Daher ist für eine neue Geometrie immer eine neue Kernberechnung notwendig, was sich als eine nichttriviale Aufgabe herausgestellt hat. Einen geschlossenen Ansatz für die Entwicklung neuer Rekonstrukti-

onskerne stellt die von Louis in [24] vorgestellte Approximative Inverse dar, die von Dietz in [6] auf die 3D-Computertomographie angewendet wurde. Sie liefert die qualitativ besten Rekonstruktionen sowohl im Vergleich mit anderen Filterentwicklungen des Typs gefilterte Rückprojektion, als auch mit anderen Rekonstruktionsverfahren.

Neben der Entwicklung von Rekonstruktionsmethoden, die den mathematischen Aspekt widerspiegeln, spielt in der 3D-Computertomographie die Umsetzung mittels numerischer Programmierung eine wesentliche Rolle. Die Faktoren dabei sind die Komplexität des Algorithmus, der von einer Vielzahl von Parametern abhängt, und die intensive Nutzung der Resourcen Rechenzeit und Speicherbedarf, bedingt durch die Menge der Daten.

Bei der Erforschung neuer Algorithmen ist es notwendig, diese schnell und bequem implementieren zu können. Während in der 2D-Computertomographie noch Rapid-Development-Umgebungen wie Matlab als Basis dienen können, ist dies in der 3D-Computertomographie nicht mehr möglich, denn der Speicherbedarf und die Rechenzeit liegen an der Grenze dessen, was auf Single-Prozessor-Systemen momentan möglich ist.

Ein praktischer Einsatz der Algorithmen, z.B. in medizinischen Scannern, bringt noch grössere Anforderungen mit sich. Die Rekonstruktionen sollten direkt nach der Datengewinnung zur Verfügung stehen. Daten und Rekonstruktionen müssen in Datenbanken verwaltet werden. Die Visualisierung der Ergebnisse ist ein eigenes Forschungsgebiet, in dem die Methoden einer ständigen Verbesserung unterworfen sind. Während in der Entwicklung von Rekonstruktionsverfahren die Priorität auf der möglichst exakten Annäherung der Rekonstruktion an die Ursprungsfunktion liegt, ist in der Praxis der diagnostische Wert das Kriterium, an dem sich die Darstellung orientiert. Anwendungsbezogene Visualisierung ist also ein wesentlicher Faktor.

In gerätetechnischer Hinsicht spielt die Datengewinnung an sich für die numerische Umsetzung eine Rolle. Die Abtasteinheit ist mechanischen Toleranzen unterworfen, was sich darin ausdrückt, dass die theoretisch erwünschten Aufnahmepositionen in der Praxis verschoben sind. Die Detektoren, mit denen gemessen wird, werden ständig weiterentwickelt und das Modell des Rekonstruktionsverfahrens muss dementsprechend modifiziert werden.

Die Speicherung der Daten findet mit einer Vielzahl von Datenformaten statt, von denen die einfachsten die rohen Daten kapseln und die Parameter getrennt vorliegen; komplexere, wie das DICOM-Verfahren, speichern geometrische und physikalische Informationen mit den Daten zusammen ab.

Die Zielsetzung dieser Arbeit, welche an der Schnittstelle von angewandter Mathematik und Informatik angesiedelt ist, umfasst die Behandlung zweier Probleme in der 3D-Computertomographie. Zum einen die Entwicklung eines Rekonstruktionsverfahrens mit einer **G**eometrie-**U**nabhängigen Strategie der **F**ilterentwicklung, kurz GUF-Verfahren, welches eine Alternative zu den üblichen, geometrieabhängigen Rekonstruktionsmethoden bietet. Zum anderen die Definition mathematischer Muster und Implementierung eines C++–Klassenframeworks, welches die Notwendigkeiten der praktischen Umsetzung berücksichtigt. Letzters wird im folgenden TORNE-Framework genannt, wobei der Name ein Akronym für **TO**mographic–**R**econstruction–**NE**twork ist.

Die Arbeit gliedert sich in drei Teile. Im ersten Teil wird das GUF-Verfahren entwickelt. Dieses ist ein Hybridverfahren aus gefilterter Rückprojektion und Filterung nach Rückprojektion. Die Idee ist, statt f eine Approximation

$$\widetilde{f}_\Gamma = H_\Gamma * \mathcal{D}_\Gamma^*(\mathcal{D}_\Gamma f * \mathcal{D}_\Gamma e)$$

an die geglättete Funktion

$$\widetilde{f} = f * e$$

zu bestimmen, wobei Γ die Aufnahmegeometrie darstellt.
Der Filterkern H_Γ wird aus $\mathcal{D}_\Gamma^*\mathcal{D}_\Gamma e$ und der exakten Funktion e unter Verwendung verschiedener Regularisierungen berechnet. Ziel war es, ein volldiskretes Verfahren zu entwickeln, bei dem die Geometrieabhängigkeit dadurch umgangen wird, dass man $\mathcal{D}_\Gamma^*\mathcal{D}_\Gamma e$ innerhalb des verwendeten geometrischen Settings diskret berechnet und somit alle Geometrieparameter implizit berücksichtigt. Die Funktion e wird als Basisfunktion bezeichnet, da $\widetilde{f}$ von einer, durch diese Funktion generierten Basis aufgespannt wird. Als Basisfunktionen e werden Funktionen untersucht, die sich zur Darstellung besonders eignen. Eine Bedingung an

die Wahl dieser Funktionen ist, dass ihre Röntgen-Transformation analytisch anzugeben ist.

Der Aufbau dieses Teils ist wie folgt. Im ersten Kapitel werden die physikalischen Gegebenheiten bei der Datengewinnung beschrieben und ein mathematisches Modell entwickelt. Ausserdem werden die verwendeten Operatoren definiert.

Das zweite Kapitel enthält eine Zusammenfassung der wichtigsten Ergebnisse von [6]. Innerhalb dieses theoretischen Kontextes ist die Geometrieabhängigkeit der Verfahren des Typs "gefilterte Rückprojektion" direkt zu erkennen.

Das dritte Kapitel stellt das GUF-Verfahren in den Mittelpunkt. Nachdem am Anfang noch einmal im Detail der Ansatz motiviert wird, folgt erst eine analytische Darstellung der Einzelschritte. Danach werden diese, da es sich um ein diskretes Verfahren handelt, in einem diskreten Kontext unter Verwendung aller Geometrieparameter konkretisiert. Den Abschluss des Kapitels bildet die Darstellung der Röntgen-Transformation aller verwendeten Basisfunktionen für die Voruntersuchung im zweidimensionalen und die Hauptuntersuchung im dreidimensionalen Fall.

Das vierte Kapitel zeigt die Rekonstruktionsergebnisse in 2D und 3D. Das GUF-Verfahren wurde an verschiedenen Arten von Aufnahmegeometrien getestet. Diese besitzen teilweise praktische Relevanz, teilweise haben sie einen Benchmark-Charakter. Sie umfassen eine Approximation an vollständige Daten, verschiedene Tuy–Kirillov–Geometrien und zwei kreisförmige Abtastkurven. Hier sind auch Rekonstruktionen realer Daten mit Vergleich zur Rekonstruktion mittels Approximative Inverse zu finden.

Der zweite Teil dieser Arbeit entwickelt das TORNE-Framework. Die verschiedenen Rekonstruktionsverfahren der 3D-Computertomographie werden abstrahiert und man gelangt zu einem Satz mathematischer Muster, analog zu den Entwurfsmustern, die im Softwaredesign verwendet werden. Diese Muster beschreiben die Objekte und Prozesse, die beim Implementieren inverser Probleme, speziell der Computertomographie, auftreten. Von diesen Mustern wird ein Satz von Objekten abgeleitet, die zur Modellierung von Rekonstruktionsverfahren dienen. Richtlinien beim Design der Muster und der Objekte sind u.a. schnelle

Entwicklungszyklen, eine leichte Anbindung von verschiedenen Datenquellen und Ausgabemöglichkeiten, letztere im Sinne von Datenspeicherung und Visualisierung. Die Objekte sind, wie der Name des Frameworks schon sagt, in Form eines Netzwerkes mit impliziter Ablaufkontrolle verbunden, da dieser Ansatz die grösstmögliche Flexibilität bietet. Mit Hilfe des abstrakten Rahmens, den die mathematischen Muster bieten, und der innerhalb dieses Kontextes entwickelten Objekte sind die oben genannten Probleme der Implementierung angemessen zu lösen.
Der Aufbau des zweiten Teils ist wie folgt. Im ersten Kapitel wird eine Motivation des Frameworks gegeben.
Das zweite Kapitel umfasst eine Einführung in die Objektorientierte Programmierung (OOP). Diese ist notwendig, da die OOP noch nicht zum Standardrepertoire numerischer Programmierung gehört. Weiterhin werden hier Fragen des allgemeinen Designs erläutert und Entwurfsmuster als Abstraktion von Objekten definiert.
Das dritte Kapitel stellt Fragen des konkreten Designs von TORNE in den Mittelpunkt. Die mathematischen Muster werden hergeleitet und Implementierungsaspekte betrachtet, die beim Ableiten von Objekten aus dem Musterkatalog wichtig sind. Abschliessend werden Ablaufkontrolle und Objektschnittstellen konkretisiert.
Das vierte Kapitel zeigt an drei Beispielen die Implementierung von Rekonstruktionsalgorithmen im TORNE-Framework. Als Beispiele wurden die gefilterte Rückprojektion mittels Approximativer Inverse, Tomosynthese und das GUF-Verfahren gewählt. Die ersten beiden Verfahren wurden gewählt, um zu zeigen, dass, wenn die zeitaufwendigen analytischen Vorberechnungen für den Filter durchgeführt wurden, die eigentliche Implementierung innerhalb von TORNE nur minimale Änderungen erfordert.

Den dritten Teil bildet der komplette Muster- und Objektkatalog der aktuellen Version von TORNE. Dieser Teil stellt eine Referenz für die Implementierung dar.

Mein herzlicher Dank geht zuerst an Prof. Dr. A.K. Louis für das vielfältige Wissen, das er mir vermittelt hat und dafür, dass er mir Raum gegeben hat, meine Ideen zu entwickeln. Weiterhin danke ich allen Mitarbeitern der Arbeitsgruppe für die freundschaftliche Atmosphäre, die viel zur Produktivität beigetragen hat und insbesondere Dr. Thomas Schuster, Dr. Uwe Schmitt, Jens Mohr und Achim Domma für die inspirierenden Diskussionen.
Last but not least danke ich meiner Schwester Eva Müller und meinen Eltern für ihre Unterstützung und meiner lieben Géraldine für ihre beständige Aufmunterung, ohne die diese Arbeit wohl so nicht zustande gekommen wäre.

Kapitel 1

Eine neue Rekonstruktionsstrategie

1.1 Grundlagen der 3D-Computer-Tomographie

1.1.1 Problemstellung und Verfahrensbeschreibung

Die 3D-Computer-Tomographie ist ein bildgebendes Verfahren, bei dem Informationen über das Innere eines dreidimensionalen Untersuchungsobjektes mit Hilfe zweidimensionaler Projektionen gewonnen werden. Die Anwendungen reichen von der Medizintechnik, bei der das "Untersuchungsobjekt" Patienten sind, über Kristallographie, bei der Informationen über die Gestalt eines Kristalles gesucht werden bis hin zum zerstörungsfreien Prüfen, bei dem z.B. Werkstücke auf Risse überprüft werden.
Das grundlegende Verfahren ist in allen Fällen das gleiche:
Das Untersuchungsobjekt wird von einer Seite, ausgehend von einer *Quelle* mit Röntgenstrahlung bestrahlt, deren Abminderung auf der anderen mit Hilfe eines *Detektors* gemessen wird. Dann werden Quelle und Detektor zu einer anderen Position bewegt und wieder gemessen, bis eine ausreichende Anzahl Daten vorliegt. Aus diesen wird nun eine dreidimensionale Darstellung des Untersuchungsobjektes errechnet.
Die Rahmenbedingungen der Datengewinnung hängen vom Einsatzgebiet ab:
In medizinischen Anwendungen ist es notwendig, die Strahlenbelastung für den Patienten so gering wie möglich zu halten, d.h. es werden wenige Projektionen gemessen und die

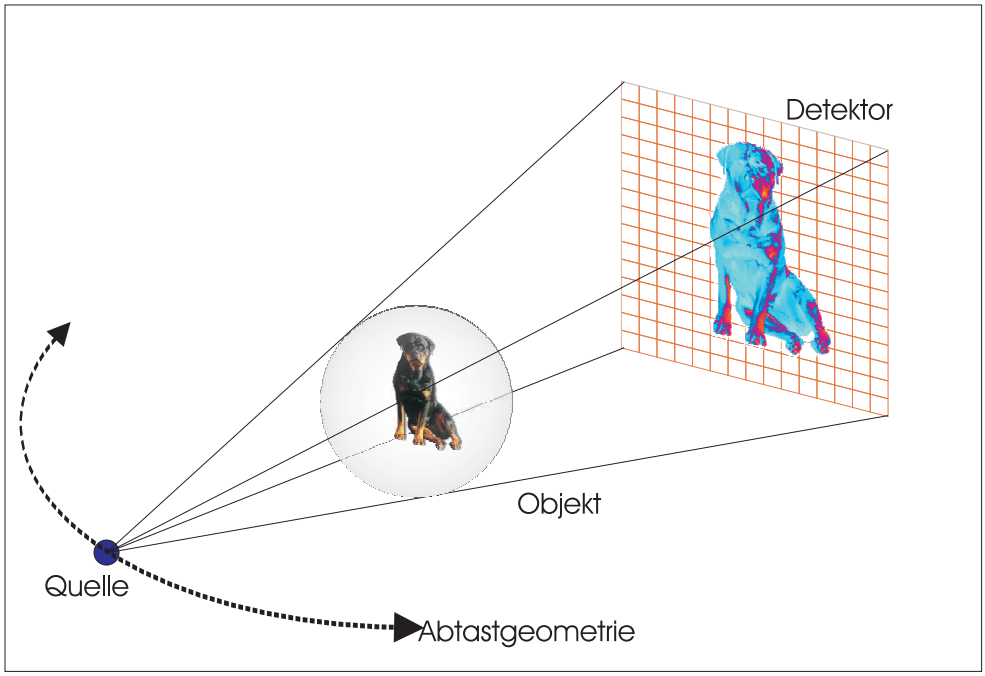

Abbildung 1.1: Datengewinnung

Intensität der Röntgenstrahlung pro Projektion minimiert. Ausserdem muss die Geometrie so angeordnet sein, dass die Messzeit kurz ist, um inkonsistente Daten zu vermeiden, die durch Bewegung des Patienten entstehen würden (ein alternativer Zugang dazu ist die Modellierung als zeitabhängiges Problem. Siehe [38]).

Beim zerstörungsfreien Prüfen ist die Strahlung von der Empfindlichkeit des Untersuchungsobjektes abhängig. Im Allgemeinen kann hier mit härterer Strahlung, d.h. Strahlung höherer Energie, und mehr Projektionen gemessen werden. Wichtig ist bei der Bestimmung der Strahlung, dass der Kontrast auf dem Detektor optimal ist. Zu harte Strahlung wird kaum gedämpft und führt zu sogenannten Spitzlichtern. Bei diesen wird der Maximalwert messbarer Strahlung auf dem Detektor erreicht oder sogar überschritten. Zu geringe Strahlung manifestiert sich in kontrastarmen Bildern, die ein wesentlich schlechteres Signal/Rausch-Verhältnis aufweisen, da das Rauschen gerade bei niedrigen Detektorwerten lokalisiert ist.

Grundlage der Modellierung ist die Betrachtung eines einzelnen Röntgenstrahls, der mit

der Anfangsintensität I_0 von der Quelle ausgeht, das Objekt durchdringt, dabei eine Abminderung erfährt und auf dem Detektor mit der Intensität I_L auftrifft. Physikalisch spielen hier ausser der Anfangsintensität die Strahlhärte und die Verwendung mono- oder polychromatischer Strahlung, d.h. Strahlung mit einer oder mehrerer Wellenlängen eine Rolle. Bei der Modellierung, die wir hier verwenden, sei monochromatische Strahlung angenommen. Ausserdem setzen wir voraus, dass sich die Strahlen geradlinig ausbreiten. Der Strahl erfährt bei der Durchdringung des Objektes nun eine exponentielle Dämpfung, die zu

$$I_L = I_0 \mathrm{e}^{-\int_L \mu(t)dt}$$

führt, mit L, dem Strahl durch das Objekt, und μ, dem Absorbtionskoeffizienten, der wiederum proportional zur Dichte ist. Dadurch gelangt man zu der Darstellung

$$\int_L \mu(t)dt = -\ln(I_L/I_0).$$

Das inverse Problem besteht nun darin, aus allen gemessenen Strahlen die Absortionskoeffizienten μ zu bestimmen. In der folgenden Arbeit wird die gesuchte Funktion f mit ihren Absorbtionskoeffizienten identifiziert. Eine mögliche Vorgehensweise, die später genauer dargestellt ist, wird es sein, die Problemlösung in zwei Schritten durchzuführen. Der erste ist eine Filterung der Daten, der zweite die Anwendung des adjungierten Operators der Modellierung, der Rückprojektion genannt wird.

Wenden wir uns der Geometrie, die einen entscheidenden Einfluss auf das inverse Problem hat, zu. Der Gantry, also die Abtasteinheit, bestehend aus Röntgenquelle, Detektor und mechanischem Aufbau, bestimmt die Geometrie, welche die Gesamtheit der in die mathematische Modellierung eingehenden Parameter ist. Siehe dazu [15].

Die Röntgenquelle setzt den Rahmen der Modellierung, denn die Strahlanordnung hat einen entscheidenden Einfluss auf die Auswahl der verwendeten Operatoren. In allen praktischen Belangen wird eine punktförmige Quelle eingesetzt, die, wie wir später sehen, zur divergenten Röntgentransformation als mathematisches Modell führt. Hier geht von einem Punkt ein Bündel von Strahlen aus, die das Objekt in Form eines Kegels überdecken, der auf einem rechteckigen Detektor gemessen wird. Diese Anordnung wird daher auch "Cone–Beam–Anordnung" genannt. Aus mathematischen Gründen wäre eine parallele Strahlanordnung

wünschenswert, bei der mit jedem Punkt auf dem Detektor ein senkrecht auftreffender, einzelner Strahl der Quelle korreliert, was eine flächige Quelle notwendig machen würde. Diese, praktisch nicht realisierbare Anordnung, würde bei der Modellierung zur parallelen Röntgen-Transformation führen, die mathematisch angenehme Eigenschaften, wie zum Beispiel die im Fourier-Slice-Theorem besagten, besitzt (siehe Abschnitt 1.3.1). Die bei der Modellierung oft verwendete RADON-Transformation ist bezüglich Inversion und Genauigkeitsabschätzungen die am besten untersuchte, würde aber einen Gantry erfordern, der über ganze Schichten, statt über Strahlen misst, was technisch nicht möglich ist. Glücklicherweise kann man mit Hilfe der Formel von Grangeat einen Zusammenhang zwischen RADON-Transformation und divergenter RÖNTGEN-Transformation herstellen, so dass die Ergebnisse zu übertragen sind.

Der Detektor misst, entgegen der vereinfachten mathematischen Modellierung nicht die Abminderung eines einzelnen Strahls. Jedes Detektorelement umfasst eine Fläche A und wird gepulst ausgelesen, d.h. was eigentlich gemessen wird ist

$$I_L(A,T) = \sum_T \mu_i$$

wobei T den Zeitraum eines Messvorgangs darstellt und μ_i die Dämpfung der Strahlen, die innerhalb von T auf dem Detektorelement A auftreffen. Es wird somit eine Integration über ein Teilvolumen des Objektes approximiert. Die Ausgangsintensität I_0 wird praktisch durch eine Messung ohne Objekt bestimmt (was gleichzeitig eine Abschätzung des Rauschens erlaubt) und stellt somit ebenfalls eine Summe von Photonen gemessen über den Zeitraum T dar. Sie kann ansonsten auch durch das Maximum des Wertebereichs des Detektors approximiert werden. Diese Abweichungen vom Modell haben jedoch keinen Einfluss auf die Anwendbarkeit desselbigen.

Die Führung des Gantrys bestimmt nun das, was oft als Abtastgeometrie bezeichnet wird, auch wenn es nur einen Teil der Geometrie darstellt: die Gesamtheit der Positionen von Quelle und Detektor im dreidimensionalen Raum. Lassen wir den Detektor ausser acht, für dessen Position nur wünschenswert ist, dass der Mittelstrahl des Kegels senkrecht auf die Mitte des Detektors fällt, so bewegt sich, ein feststehendes Objekt vorausgesetzt, die

Quelle auf einem Weg $\Gamma \subset \mathbb{R}^3$ im Raum. Zur Filterberechnung wird, wie wir später sehen werden, dieser Weg als analytisch beschreibbar vorausgesetzt. Er wird z.B. als Kreis mit festem Abstand d_R von Quelle und Objektzentrum oder als Spirale modelliert. Man muss sich hier allerdings im klaren sein, dass dies eine Approximation darstellt, denn bedingt durch die mechanische Toleranz des Gantrys können je nach Apparatur Abweichungen von der idealisierten Beschreibung bis zu mehreren Prozent auftreten, die beim Rekonstruktionsprozess über eine Korrekturtabelle in die Rückprojektion eingehen, aber nicht in die Filterberechnung. Dieser Punkt ist neben analytischen Betrachtungen eine Grund für die Entwicklung des GUF-Verfahrens. GUF steht hierbei für *G*eometrie*U*nabhängige *F*ilterstrategie. Eine ausführliche Darstellung verschiedener Geometrien wird in Abschnitt 1.4.2 untersucht werden.

Für eine ausführliche Darstellung der physikalischen und mechanischen Aspekte der 3D-Computertomographie sei hier auf [15] und [11] verwiesen.

1.1.2 Das mathematische Modell

Nach der Beschreibung der Problemstellung im letzten Abschnitt wenden wir uns der mathematischen Modellierung zu. Vorher noch einige Definitionen, die im folgenden verwendet werden und die sich in ihrer Bezeichnung an [31] und [6] anlehnen:
Die Dimension der Problemstellung sei mit n bezeichnet.

Bemerkung 1.1.1. Es handelt sich bei dieser Arbeit um den Bereich der 3D-Computertomographie, so dass ohne weiteres $n = 3$ fixiert werden könnte. Da die folgenden Aussagen jedoch allgemeingültiger sind und in Abschnitt 1.3.5 auch die 2D-Computertomographie als Voruntersuchung auftaucht, ist die Verwendung dieses Parameters gerechtfertigt.

Verwendete Mengen:
$S^{n-1} = \{x \in \mathbb{R}^n : ||x||_2 = 1\}$ ist die Menge der Einheitsrichtungen
Ω^n steht für die Einheitskugel im $\mathbb{R}^n$,
$Z := S^{n-1} \times [-1, 1]$ sei der Einheitszylinder,
$\widetilde{Z} := S^{n-1} \times \mathbb{R}$ sei der in Längsachse unbeschränkte Zylinder,
$T := \{(\theta, y) : \theta \in S^{n-1}, y \in \theta^\perp \cap \Omega^n\}$ und $\widetilde{T} = \{(\theta, y) : \theta \in S^{n-1}, y \in \theta^\perp\}$ seien Tangentenbündel
Verwendete Räume
$\mathcal{S}(\mathbb{R}^n)$ Schwartz-Raum der schnell fallenden Funktionen,
$\mathsf{L}_2(\mathbb{R}^n) := \{f | \int_{\mathbb{R}^n} |f(x)|^2 dx < \infty\}$ der Raum der quadratintegrierbaren Funktionen,
$\mathsf{H}_0^\alpha(\Omega)$ Sobolev–Raum: Abschluss von $\mathsf{C}_0^\infty(\Omega)$ (Raum der beliebig oft differenzierbaren Funktionen mit kompaktem Träger in Ω) bezüglich der Norm

$$\|f\|_\alpha^2 = \int_{\mathbb{R}^n} (1 + |\xi|^2)^\alpha \, |\widehat{f}(\xi)|^2 \, d\xi, \; \alpha \geq 0,$$

in $\mathsf{L}_2(\Omega)$.
verwendete Transformationen
n-dimensionale FOURIER-Transformation:

$$\mathcal{F}_n f(\xi) = \widehat{f}(\xi) := \int_{\mathbb{R}^n} f(x) \exp(-ix \cdot \xi) dx$$

mit der Inversen

$$\mathcal{F}_n^{-1} f(x) = f^\vee(x) := (2\pi)^{-n} \int_{\mathbb{R}^n} f(\xi) \exp(ix \cdot \xi) d\xi$$

Das Riesz–Potential:

$$(I^\alpha f)^\wedge(\xi) = |\xi|^{-\alpha} \widehat{f}(\xi)$$

Mit obigen Bezeichnungen lässt sich nun das Problem mittels der folgenden Integraltransformationen definieren:

Röntgen-Transformation

Stehen die gemessenen Daten eines Strahls als

$$g_L := -\ln(I_L/I_0) = \int_L f(t) dt$$

zur Verfügung und betrachten wir die Funktion f als $f \in C_0^\infty(\Omega^n)$, so entsprechen die g_L der RÖNTGEN-Transformation von f, die für unbeschränkte Gebiete definiert ist als

$$\begin{aligned} \mathcal{P} : \mathcal{S}(\mathbb{R}^n) &\to \mathcal{S}(\widetilde{T}) \\ \mathcal{P}f(\theta, y) &= \int_{\mathbb{R}} f(y + t\theta)\, dt \end{aligned}$$

bzw. in unserem Fall mit beschränktem Gebiet

$$\begin{aligned} \mathcal{P} : \mathsf{C}_0^\infty(\Omega^n) &\to \mathsf{C}_0^\infty(T) \\ \mathcal{P}f(\theta, y) &= \int_{\mathbb{R}} f(y + t\theta)\, dt \end{aligned}$$

wobei der Strahl L als Gerade durch y mit der Richtung θ parametrisiert wird. Der adjungierte Operator dazu berechnet sich als

$$\mathcal{P}^\# g(x) = \int_{S^{n-1}} g(\theta, E_\theta x)\, d\theta$$

mit der orthogonalen Projektion $E_\theta : \mathbb{R}^n \to \theta^\perp$ auf die Ebene senkrecht θ.

Die RÖNTGEN-Transformation ist Basis des mathematischen Modells der parallelen Geometrie. Eine Projektion erhält man dadurch, dass man θ fixiert und y variiert.

Sie lässt sich stetig auf gewichtete L_2–Räume mit dem Gewicht $W_\nu^n(y) = (1 - |y|^2)^{\nu - n/2}$, $\bar{W}_\nu^n(y) = (1 - |y|^2)^{\nu - (n-1)/2}$ für $(\theta, y) \in T$ fortsetzen:

Satz 1.1.1. Sei $\nu > n/2 - 1$. Die Abbildung

$$\begin{aligned} \mathcal{P} : \mathsf{L}_2(\Omega^n,\, W_\nu^{-1}) &\to \mathsf{L}_2(T,\, \bar{W}_\nu^{-1})\ , \\ \mathcal{P}f(\theta,\, y) &= \int_{\mathbb{R}} f(y + t\theta)\, dt \end{aligned}$$

ist stetig. Ihre Adjungierte ist

$$\mathcal{P}^* g(x) = W_\nu(x) \int_{S^{n-1}} g(\theta,\, E_\theta x)\, \bar{W}_\nu^{-1}(E_\theta x)\, d\theta\ .$$

Siehe hierzu [28].

Betrachten wir statt der parallelen Geometrie die praktisch relevante Cone-Beam-Geometrie, so ist die Basis der Modellierung die

Divergente Röntgen-Transformation

Hier ändert sich die Parametrisierung des Strahls, denn

$$\begin{aligned} \mathcal{D} : C_0^\infty(\Omega^n) &\to \mathcal{S}(\mathbb{R}^n \times S^{n-1}) \\ \mathcal{D}f(a,\, \theta) &= \int_0^\infty f(a + t\theta)\, \mathrm{d}t \end{aligned}$$

mit $a \notin \Omega^n$. Der adjungierte Operator ist

$$\mathcal{D}^\# g(x) = \int_{\mathbb{R}^n} g\left(a, \frac{x-a}{|x-a|}\right) |x-a|^{1-n} \mathrm{d}a,$$

die Rückprojektion wird definiert als

$$\mathcal{D}^* g(x) = \int_{\mathbb{R}^n} g\left(a, \frac{x-a}{|x-a|}\right) \mathrm{d}a,$$

Betrachtet man f auf $C_0^\infty(\Omega^n)$, so stehen vollständige Daten schon mit $a \in \Gamma = r\, S^2$, $r > 1$ zur Verfügung. Vermöge des folgenden Lemmas kann man dann die Daten der parallelen RÖNTGEN-Transformation in die der divergenten RÖNTGEN-Transformation umwandeln, so dass auch hier eine Erweiterung auf L_2-Räume möglich ist. Siehe hierzu [6].

Lemma 1.1.1. Sei $\widetilde{w}(a,\,\theta) = 2\,(|a^\top\theta|)^{-1}\,w(E_\theta a)$ für $(a,\,\theta) \in \Gamma \times S^2$, $\Gamma = r\,S^2$, $r > 1$. Dann ist

$$\begin{aligned} U : \mathsf{L}_2(T,\,w^{-1}) &\rightarrow \mathsf{L}_2(\Gamma \times S^2,\,\widetilde{w}^{-1})\,, \\ Ug(a,\,\theta) &= g(\theta,\,E_\theta a) \end{aligned}$$

ein unitärer Operator und

$$\begin{aligned} \mathcal{D} : \mathsf{L}_2(\Omega) &\rightarrow \mathsf{L}_2(\Gamma \times S^2,\,\widetilde{w}^{-1})\,, \\ \mathcal{D}f(a,\,\theta) &= \int_0^\infty f(a+t\theta)\,dt = U\mathcal{P}f(a,\theta) = \mathcal{P}f(\theta, E_\theta a) \end{aligned} \tag{1.1}$$

stetig.

Die Daten einer Projektion gewinnt man bei der divergenten RÖNTGEN-Transformation (im Gegensatz zu parallelen) dadurch, dass man die Quellposition a festhält und θ variiert.

Bemerkung 1.1.2. Auch wenn sich $\mathcal{D}f$ von $\mathcal{P}f$ durch die Parametrisierung unterscheidet, so sind die Daten *eines Strahls* grundsätzlich die gleichen, denn durch die Beschränkung $f \in C_0^\infty(\Omega^n)$ ist

$$\int_0^\infty f(a+t\theta)dt = \int_{\mathbb{R}} f(E_\theta a + t\theta)dt\,,$$

d.h. das Integral über die "fehlende Hälfte" der Halbgeraden im linken Teil ist sowieso 0. In Abschnitt 1.3 wird zur Beschreibung $\mathcal{D}f$ verwendet, damit ersichtlich ist, wie die Strahlen zu Projektionen gebündelt werden.

Radon-Transformation

Für theoretische Untersuchung und die Angabe einer Inversionsformel ist die n–dimensionale RADON-Transformation von entscheidender Bedeutung. Sie ist für unbeschränktes Gebiet definiert als

$$\begin{aligned} \mathcal{R} : \mathsf{S}(\mathbb{R}^n) &\rightarrow \widetilde{Z} \\ \mathcal{R}f(\omega,\,s) &= \int_{\omega^\perp} f(s\omega + y)\,dy \end{aligned} \tag{1.2}$$

und für ein beschränktes Gebiet als

$$\begin{aligned} \mathcal{R} : \mathcal{S}(\Omega^n) &\to Z \\ \mathcal{R}f(\omega,\, s) &= \int\limits_{\omega^\perp \cap \Omega^n} f(s\omega + y)\, dy \end{aligned} \tag{1.3}$$

mit dem adjungierten Operator

$$\mathcal{R}^{\#} g(x) = \int\limits_{S^{n-1}} g(\theta, x \cdot \theta)\, d\theta$$

Der Zusammenhang von RADON-Transformation und RÖNTGEN-Transformation ist ersichtlich, wenn wir $\mathcal{P}_\theta f(\cdot) := \mathcal{P}f(\theta, \cdot)$ setzen und $\mathcal{R}^n$ als n-dimensionale RADON-Transformation definieren, wobei $\mathcal{R}^{n-1}$ im folgenden auf $\theta^\perp$ wirkt. Es gilt dann

$$\mathcal{R}^n f(\omega, s) = \mathcal{R}^{n-1} P_\theta f(\omega, s)$$

Siehe dazu [27].

Auch die RADON-Transformation lässt sich auf gewichtete L_2-Räume erweitern. Mit $W_\nu(x) = (1 - |x|^2)^{\nu - n/2}$, $x \in \Omega^n$ und $w_\nu(s) = (1 - s^2)^{\nu - 1/2}$, $s \in [-1,\, 1]$ als den entsprechenden Gewichten, gilt

Satz 1.1.2. Sei $\nu > n/2 - 1$. Die Abbildung

$$\begin{aligned} \mathcal{R} : \mathsf{L}_2(\Omega^n,\, W_\nu^{-1}) &\to \mathsf{L}_2(Z,\, w_\nu^{-1})\,, \\ \mathcal{R}f(\omega,\, s) &= \int\limits_{\omega^\perp \cap \Omega^n} f(s\omega + y)\, dy \end{aligned} \tag{1.4}$$

ist stetig. Ihre Adjungierte ist

$$\mathcal{R}^* g(x) = W_\nu(x) \int\limits_{S^{n-1}} g(\omega,\, x^\top \omega)\, w_\nu^{-1}(x^\top \omega)\, d\omega\,.$$

Der Beweis ist in [20] zu finden.

Zum Abschluss dieses Abschnittes seien hier zur RADON-Transformation und RÖNTGEN-Transformation noch Inversionsformeln gegeben, deren Beweis z.B. in [31] zu finden ist:

Satz 1.1.3. Für $f \in \mathcal{S}(\mathbb{R}^n)$ und $\alpha < n$ gilt:

$$f = \frac{1}{2}(2\pi)^{1-n} I^{-\alpha} \mathcal{R}^{\#} I^{\alpha-n+1} g,\ g = \mathcal{R}f$$

und

$$f = \frac{1}{|\mathbb{S}^{n-2}|}(2\pi)^{-1} I^{-\alpha} \mathcal{P}^{\#} I^{\alpha-1} g,\ g = \mathcal{P}f$$

wobei $|S^{n-1}|$ die Oberfläche der Einheitssphäre S^{n-1} darstellt und das Riesz-Potential auf die zweite Variable wirkt.

Für die divergente RÖNTGEN-Transformation im Fall $n = 3$ lautet die Inversionsformel wie folgt.

Satz 1.1.4. Für $f \in \mathsf{L}_2(\Omega^3)$ gilt:

$$f(x) = \frac{1}{2\pi} \Lambda_x \int_{S^2} \mathcal{D}f(x, \theta) \mathrm{d}\theta.$$

Dabei ist Λ_x der Laplace-Operator, der auf das Integral als eine Funktion von x wirkt.

Der Beweis für beliebiges n ist in [18] zu finden.

1.1.3 Die Approximative Inverse

Inverse Probleme sind dadurch gekennzeichnet, dass sie schlecht gestellt, also nicht stetig invertierbar, sind. Anstatt das Problem $Af = g$ zu lösen, minimiert man den Abstand $||Af - g||$ und wählt gegebenenfalls unter allen Lösungen die mit der kleinsten Norm. Diese Lösung nennt man verallgemeinerte Inverse und bezeichnet sie mit $A^{\dagger} g = f^{\dagger}$. Da diese nicht stetig ist, wenn das Bild von A nicht abgeschlossen ist, behilft man sich mit sogenannten Regularisierungen, welche stetige Approximationen T_γ an $A^{\dagger}$ darstellen: $\lim_{\gamma \to 0} T_\gamma g = A^{\dagger} g,\ g \in \mathrm{Def(A)}$. Eine allgemeine Behandlung inverser Probleme findet sich in [21]. Eine ganze Klasse von Regularisierungen linearer, schlecht gestellter Probleme stellt die Approximative Inverse dar, die in [24] vorgestellt wird. Nachfolgend sei eine Zusammenfassung der Approximativen Inversen und einiger wichtiger Eigenschaften gegeben.

Es sei $\mathrm{A} : X \to Y$ ein linearer, kompakter Operator zwischen Hilberträumen X und Y. Ist $\dim(\mathrm{range}(\mathrm{A})) = \infty$, so ist das Problem schlecht gestellt. Für die folgende Arbeit ist die Einschränkung auf L_2-Räume notwendig. Daher nehmen wir $X := \mathsf{L}_2(\Omega)$ an, wobei Ω ein beschränktes Gebiet sei. Das Problem besteht nun darin, für Daten $g \in Y$ ein $f \in X$ mit $\mathrm{A}f = g$ zu finden.

Ziel ist es nun, eine geglättete Funktion

$$\widetilde{f}(x) = \langle f,\, e_\gamma(x,\cdot)\rangle_X,\ \ x \in \Omega,$$

statt f zu finden. Die Funktion e heisst Mollifier und muss den Mittelwert 1 besitzen sowie für $\gamma \to 0$ die Delta-Distribution im schwachen Sinn approximieren. Aus Notationsgründen sei ab jetzt $e = e_\gamma$. Existiert ein $\psi(x;\cdot) \in Y$ mit $\mathrm{A}^*\psi(x;\cdot) = e(x,\cdot)$, so gilt

$$\begin{aligned}\langle f,\, e(x,\cdot)\rangle_X &= \langle f,\, \mathrm{A}^*\psi(x,\cdot)\rangle_X \\ &= \langle \mathrm{A}f,\, \psi(x,\cdot)\rangle_Y \\ &= \langle g,\, \psi(x,\cdot)\rangle_Y\ ,\end{aligned}$$

Man kann also $\widetilde{f}$, die geglättete Lösung, als Skalarprodukt eines, unabhängig von den Daten zu berechnenden, Rekonstruktionskerns ψ mit f gewinnen. Falls der Mollifier $e(x,\cdot) \notin \mathrm{range}(\mathrm{A}^*)$, so soll $e \in \mathrm{range}(A^*) \oplus \mathrm{range}(A^*)^\perp$ sein. Dadurch wird eine gewisse Glattheit gewährleistet.

Durch Lösen der Normalgleichung

$$\mathrm{A}\mathrm{A}^*\psi(x;\cdot) = \mathrm{A}e(x,\cdot)$$

wird der Defekt

$$\|\mathrm{A}^*\psi(x;\cdot) - e(x,\cdot)\|_X\ ,$$

minimiert.

Der Rekonstruktionskern mit minimaler Norm entspricht der verallgemeinerten Lösung $\psi(x;\cdot) = (\mathrm{A}^*)^\dagger e(x,\cdot)$ von $\mathrm{A}^*\psi(x;\cdot) = e(x,\cdot)$. Auch dazu siehe [21].

Nach diesen Vorüberlegungen lautet die Definition der Approximativen Inversen nun:

Definition 1.1.1. Die Abbildung $\mathcal{S} : Y \to X$ mit $\mathcal{S}g(x) = \langle g,\, \psi(x;\cdot)\rangle$ heisst *Approximative Inverse* von A, die Lösung $\psi(x;\cdot)$ der Normalgleichung bezeichnet man als *Rekonstruktionskern* bezüglich des Rekonstruktionspunktes x.

Zur Verdeutlichung des Konzeptes sei hier kurz die Wirkung der Approximativen Inversen auf die Singulärwertzerlegung betrachtet.

Definition 1.1.2. Für den kompakten Operator A existiert eine Darstellung der Form

$$\mathrm{A}f = \sum_n \sigma_n \,\langle f,\, v_n\rangle_X \, u_n \ ,$$

wobei $(v_n) \subset X$ ein vollständiges Orthonormalsystem (VOS) für $\mathrm{N(A)}^\perp$ ist, $(u_n) \subset Y$ ein VOS für $\overline{\mathrm{range(A)}}$ und σ_n die Eigenwerte von $(AA^*)^{1/2}$. Dann heisst $\{v_n,\, u_n;\, \sigma_n\}_n$ *Singulärwertzerlegung* von A, die σ_n heissen *Singulärwerte* von A.
Der adjungierte Operator A^* kann dann dargestellt werden als

$$\mathrm{A}^* g = \sum_n \sigma_n \,\langle g,\, u_n\rangle_Y \, v_n \ .$$

Es gilt $\mathrm{A}v_n = \sigma_n u_n$ und $\mathrm{A}^* u_n = \sigma_n v_n$.
Mit obiger Definition stellt sich der Rekonstruktionskern dar als

$$\psi(x;\cdot) = \sum_{\sigma_n > 0} \sigma_n^{-1} \,\langle e(x,\cdot),\, v_n\rangle_X \, u_n \ . \tag{1.5}$$

Man kann die Approximative Inverse als Glättung der Pseudoinversen

$$\mathrm{A}^\dagger g = \sum_{\sigma_n > 0} \sigma_n^{-1} \,\langle g,\, u_n\rangle \, v_n$$

auffassen:

$$\mathcal{S}g = \langle e,\, \mathrm{A}^\dagger g\rangle.$$

Zu den Details siehe [26].
Wichtig für die Effizienz des Verfahrens sowohl was die Laufzeit, als auch die Speicheranforderungen angeht, ist es, nicht für jeden Rekonstruktionspunkt einen neuen Rekonstruktionskern berechnen zu müssen. Ein Vorteil der Approximativen Inversen ist es, dass sich Invarianzen des Operators auf den Rekonstruktionskern übertragen.

Lemma 1.1.2. Seien $\mathrm{A} : X \to Y$, $T_1^x : X \to X$, $T_2^x : Y \to Y$ und $T_3^x : Y \to Y$ lineare, stetige Operatoren mit

$$T_1^x \mathrm{A}^* = \mathrm{A}^* T_2^x$$

und

$$T_3^x \mathrm{A} = \mathrm{A} T_1^x.$$

Weiterhin seien $e(x,\cdot)$, $E \in \mathrm{D}((\mathrm{A}^*)^\dagger)$ Mollifier mit $e(x,\cdot) = T_1^x E(\cdot)$ und ϕ Lösung von

$$\mathrm{A}\mathrm{A}^* \phi = \mathrm{A}E.$$

Dann löst

$$\psi(x) = T_2^x \phi$$

die Gleichung

$$\mathrm{A}\mathrm{A}^* \psi(x) = \mathrm{A}e(x,\cdot).$$

Für $e(x,\cdot)$, $E \in \mathrm{range}(\mathrm{A}^*)$ und ϕ Lösung von

$$\mathrm{A}^* \phi = E$$

gilt äquivalent:

$$\psi(x) = T_2^x \phi$$

ist Lösung von

$$\mathrm{A}^* \psi(x) = e(x,\cdot).$$

Beweis: Es gilt:

$$\mathrm{A}\mathrm{A}^* \psi(x) = \mathrm{A} T_1^x E = T_3^x \mathrm{A}E = T_3^x \mathrm{A}\mathrm{A}^* \phi = \mathrm{A}\mathrm{A}^* T_2^x \phi$$

und

$$\mathrm{A}^* \psi(x) = T_1^x E = T_1^x \mathrm{A}^* \phi = \mathrm{A}^* T_2^x \phi,$$

(siehe auch [24]). ∎

1.2 Inversionsformeln und Rekonstruktion

1.2.1 Anwendung der AI auf die Computertomographie: Inversion und Filterberechnung

Im folgenden Abschnitt wird das Konzept der Approximativen Inversen auf die relevanten Integraltransformationen der Computer-Tomographie angewendet. Es handelt sich um eine Zusammenfassung der wichtigsten Ergebnisse von [6], wo weitere Details und Beweise zu finden sind. Zu jeder Transformation werden die Invarianzen angegeben, die die Kernberechnung vereinfachen und die Inversionsformel dargestellt. Die Notation der Invarianzen entspricht Lemma 1.1.2.

Die Inversionsmethode mittels der Approximativen Inversen , welche vom Typ "gefilterte Rückprojektion" ist, liefert insbesondere mit realen Daten im Vergleich z.B. zum Feldkamp-Algorithmus (siehe [7]) die besten Ergebnisse und ist daher "state of the art". Daher wird eine ausführliche Darstellung gegeben.

1.2.1.1 RADON-Transformation

Invarianzen

Um die die folgenden Aussagen treffen zu können, ist es notwendig, das Gewicht der RADON-Transformation (siehe Satz 1.1.2) mit $\nu = n/2$ zu wählen, also $W_{n/2}(x) = 1$ und $w_{n/2}(s) = (1-s^2)^{(n-1)/2}$

Mit dieser Vorbedingung gelten dann folgende Invarianzen:

Translationsinvarianz

Vermöge der Definitionen

$$\begin{aligned} T_1^x : \mathsf{L}_2(\Omega^n) &\rightarrow \mathsf{L}_2(\Omega^n) \,, \\ f(\cdot) &\mapsto 2^{-n} f\left(\frac{\cdot - x}{2}\right) \end{aligned}$$

und

$$T_2^x : \mathsf{L}_2(Z,\, w^{-1}) \rightarrow \mathsf{L}_2(Z,\, w^{-1}) \,,$$

$$g(\omega,\, s) \quad \mapsto \quad 2^{-n}\, g(\omega,\, \frac{s - x^\top \omega}{2})\, w^{-1}\left(\frac{s - x^\top \omega}{2}\right)\, w(s)$$

gilt

$$T_1^x\, \mathcal{R}^* g(y) = \mathcal{R}^* T_2^x\, g(y).$$

Wählt man einen Mollifier, der $e(x,\, y) = T_1^x e^0(y)$ erfüllt, so muss der Rekonstruktionskern nur für $x = 0$ ausgerechnet werden. Ein solcher wird im folgenden mit ψ^0 bezeichnet.

Rotationsinvarianz

Sei U eine unitäre Transformation

$$T_1^U f(y) := f(Uy) \tag{1.6}$$

und

$$T_2^U g(\omega,\, s) := g(U\omega,\, s)\ . \tag{1.7}$$

Dann gilt

$$T_1^U \mathcal{R}^* g(y) = \mathcal{R}^* T_2^U g(y).$$

Somit muss der Rekonstruktionskern nur noch für eine Richtung berechnet werden. Für rotationssymmetrische Mollifier ist er von der Richtung unabhängig und wird als $\psi^0(s) := \psi^0(\omega,\, s)$ bezeichnet.

Dilatationsinvarianz

Mit $\varrho \geq 1$ sei

$$T_1^\varrho f(x) := \varrho^{-n}\, f(x/\varrho) \tag{1.8}$$

und

$$T_2^\varrho g(\omega,\, s) := \varrho^{-n}\, g(\omega,\, s/\varrho)\, w^{-1}(s/\varrho)\, w(s)\ . \tag{1.9}$$

Dann gilt

$$T_1^\varrho \mathcal{R}^* g(y) = \mathcal{R}^* T_2^\varrho g(y)$$

Somit übertragen sich Dilatationseigenschaften des Mollifiers auf den Kern.

Unter Berücksichtigung der Invarianzen kann man nun folgenden Satz angeben:

Satz 1.2.1. Sei $\varrho \geq 1$, U eine unitäre Transformation und $e(x,\cdot)$, e^ϱ seien Mollifier mit

$$e(x,\, y) = (2\varrho)^{-n}\, e^\varrho(U\left(\frac{y-x}{2\varrho}\right)\,. \tag{1.10}$$

Dann gilt für den Rekonstruktionskern $\psi(x)$ als Lösung von $\mathcal{R}^*\psi(x;\cdot) = e(x,\cdot)$ die Gleichung

$$\psi(x;\,\omega,\,s) = (2\varrho)^{-n}\,\psi^{0,\,\varrho}\left(\frac{s-x^\top\omega}{2\varrho}\right)\,w^{-1}\left(\frac{s-x^\top\omega}{2\varrho}\right)\,w(s)\,,$$

wobei $\psi^{0,\varrho}$ die Lösung von $\mathcal{R}^*\psi^{0,\,\varrho} = e^\varrho$ ist.

Ausserdem folgt direkt

Korollar 1.2.1. Wird der Rekonstruktionskern $\psi(x)$ mit $\mathcal{R}^*\psi(x;\cdot) = e(x,\cdot)$ aus einem Mollifier $e(x,\cdot)$ berechnet, der (1.10) erfüllt, dann lautet die Approximative Inverse der Radon–Transformation

$$\mathcal{S}g(x) = (2\varrho)^{-n} \int\limits_{S^{n-1}} \int\limits_{-1}^{1} g(\omega,\, s)\,\psi^{0,\,\varrho}\left(\frac{s-x^\top\omega}{2\varrho}\right)\,w^{-1}\left(\frac{s-x^\top\omega}{2\varrho}\right)\,ds\,d\omega\,.$$

Inversionsformel

Wenn die Radon–Transformation zur Abbildung auf $\mathsf{L}_2(\widetilde{Z})$ erweitert wird, so gilt für die Inversionsformel für $e(x,\cdot) \in \mathsf{H}_0^\alpha(\Omega^n)$, $\alpha \geq (n-1)/2$,

$$e(x,\cdot) = \frac{1}{2}\,(2\pi)^{1-n}\,\mathcal{R}^\#\,I^{1-n}\,\mathcal{R}e(x,\cdot), \tag{1.11}$$

siehe [31]. Mit Hilfe der Inversionformel lässt sich direkt ein Rekonstruktionskern angeben:

$$\psi(x) = \frac{1}{2}\,(2\pi)^{1-n}\,I^{1-n}\,\mathcal{R}e(x,\cdot).$$

Die Invarianzen übertragen sich wieder.

Satz 1.2.2. Sei $\varrho \geq 1$, U eine unitäre Transformation und $e(x,\cdot)$, $e^\varrho \in \mathsf{H}_0^\alpha(\Omega^n)$, $\alpha \geq (n-1)/2$, seien Mollifier mit

$$e(x,\, y) = (2\varrho)^{-n}\, e^\varrho(U\left(\frac{x-y}{2\varrho}\right)\,.$$

Dann gilt für den Rekonstruktionskern ψ als Lösung von $\mathcal{R}^\#\psi = e^\varrho$ die Gleichung

$$\psi(x;\,\omega,\,s) = (2\varrho)^{-n}\,\psi^{0,\,\varrho}\left(\frac{s-x^\top\omega}{2\varrho}\right)\,,$$

wobei $\mathcal{R}^\#\psi^{0,\,\varrho} = e^\varrho$ gilt und $w \equiv 1$.

Da das RIESZ-Potential in ungeraden Dimensionen die Ableitung darstellt, gilt in dem für diese Arbeit interessanten Fall mit $n = 3$:

$$\psi^0(s) = -\frac{1}{8\pi^2}\,\mathcal{R}_\omega e''(s) \tag{1.12}$$

1.2.1.2 RÖNTGEN-Transformation

Invarianzen

Unter Verwendung der, ähnlich den sich bei der RADON-Transformation ergebenden, Invarianzen

$$T_1^x f(z) := \frac{1}{8}\, f\left(\frac{z-x}{2}\right) \text{ und } T_2^x g(\theta,\, y) := \frac{1}{8}\, g\left(\theta,\, \frac{y - E_\theta x}{2}\right)\, w^{-1}\left(\frac{y - E_\theta x}{2}\right)\, w(y)\ , \tag{1.13}$$

und

$$T_1^U f(z) := f(Uz) \text{ und } T_2^U g(\theta,\, y) := g(U\theta,\, Uy), \tag{1.14}$$

T^ϱ analog 1.8, kann man sich wieder auf die Berechnung eines, von der Richtung unabhängigen, Rekonstruktionskerns für $x = 0$ beschränken. Es ergibt sich, äquivalent zu Satz 1.2.1:

Satz 1.2.3. Sei U eine unitäre Transformation und $e(x, \cdot)$ sei ein Mollifier mit

$$e(x,\, y) = \frac{1}{8}\, e(0,\, U\left(\frac{x-y}{2}\right)) = \frac{1}{8}\, e^0(U\left(\frac{x-y}{2}\right))\ .$$

Dann gilt für die Lösung von $\mathcal{P}^*\psi(x) = e(x, \cdot)$ die Gleichung

$$\psi(x;\, \theta,\, y) = \frac{1}{8}\, \psi^0\left(\frac{y - E_\theta x}{2}\right)\, w^{-1}\left(\frac{y - E_\theta x}{2}\right)\, w(y)\ ,$$

wobei ψ^0 den von der Richtung unabhängigen Rekonstruktionskern für $x = 0$ bezeichnet, d. h. $\mathcal{P}^*\psi^0 = e^0$.

Inversionsformel

Wählt man $n = 3$ und erweitert die RÖNTGEN-Transformation auf $\mathsf{L}_2(\widetilde{T})$, so kann man mittels der Inversionsformel der parallelen RÖNTGEN-Transformation , die u.a. in [31] zu finden ist, den Rekonstruktionskern angeben als

$$\psi(x) = (2\pi)^{-2} I^{-1} \mathcal{P} e(x, \cdot).$$

Dieser Rekonstruktionskern löst wiederum

$$\mathcal{P}^{\#}\psi(x) = e(x,\cdot) \text{ mit } e(x,\cdot) \in H_0^{\alpha}(\Omega^n),\ \alpha \geq 1/2$$

Unter Verwendung der Invarianzen ist

$$\psi^0(y) = (2\pi)^{-5/2} \int\limits_{\theta^{\perp}} |\eta|\, \widehat{e^0}(\eta)\, \exp(\imath\, y^{\top}\eta)\, d\eta\ .$$

Somit ergibt sich die Approximative Inverse als

$$\mathcal{S}g(x) = \int\limits_{S^2} \int\limits_{\theta^{\perp}\cap\Omega^n} g(\theta,\, y)\, \psi^0(y - E_\theta x)\, w^{-1}(y)\, dy\, d\theta \tag{1.15}$$

mit $\mathcal{P}^*\psi^0 = e^0$, bzw.

$$\mathcal{S}g(x) = \int\limits_{S^2} \int\limits_{\theta^{\perp}} g(\theta,\, y)\, \psi^0(y - E_\theta x)\, dy\, d\theta$$

mit $\mathcal{P}^{\#}\psi^0 = e^0$.

1.2.1.3 Divergente RÖNTGEN-Transformation

Bei der divergenten RÖNTGEN-Transformation ist die Datenlage von entscheidender Bedeutung für die Inversionsformel. Sie wird bestimmt durch die Aufnahmegeometrie und unterteilt sich in vollständige Daten, TUY–KIRILLOV–Daten (siehe 1.2.1) und Daten aus kreisförmiger Abtastung. Während man zu den ersten beiden Inversionsformeln angeben kann, ist dies bei letzterer Geometrie nicht mehr ohne weiteres möglich. Es wird eine Approximation angegeben, die im Kontext der Approximativen Inversen dargestellt ist.

Vollständige Daten

Im Falle vollständiger Daten ist der Rekonstruktionskern aus dem der parallelen RÖNTGEN-Transformation zu gewinnen. Der Zusammenhang zwischen diesen Transformationen ist, wie schon erwähnt,

$$\mathcal{D}f(a,\, \theta) = \mathcal{P}f(\theta,\, E_\theta a)\ ,$$

wobei $E_\theta a$ die Projektion von a auf die Ebene senkrecht θ ist.

Das Lemma 1.1.1 erlaubt es, den Rekonstruktionskern und die Approximative Inverse für die divergente RÖNTGEN-Transformation anzugeben.

Satz 1.2.4. Sei $\psi_{\mathcal{D}}$ der Rekonstruktionskern für die Abbildung (1.1), $\psi_{\mathcal{P}}$ der Rekonstruktionskern für die parallele Röntgen–Transformation und $\psi^0_{\mathcal{P}} = \psi_{\mathcal{P}}(0;\cdot)$. Dann gilt

$$\psi_{\mathcal{D}}(x;\, a,\, \theta) = \psi^0_{\mathcal{P}}(E_\theta(a-x))\ ,$$

und die Approximative Inverse von $\mathcal{D}$ lautet

$$\mathcal{S}g(x) = \int_\Gamma \int_{S^2} g(a,\, \theta)\, \psi^0_{\mathcal{P}}(E_\theta(a-x))\, \widetilde{w}^{-1}(a,\, \theta)\, d\theta\, da$$

Der Beweis findet sich in [6].

Tuy-Kirillov-Kurven

Definition 1.2.1. Eine Abtastkurve Γ heisst TUY–KIRILLOV–*Kurve*, wenn jede Ebene, die die Einheitskugel Ω^n schneidet, Γ in mindestens einem Punkte schneidet, d.h. für alle $(\omega,\, s)\,, \omega \in S^2\,, s \in [-1,\, 1]$ existiert ein $a \in \Gamma$, so dass $a^\top \omega = s$ und $a'^\top \omega \neq 0$. Dabei ist a' der Richtungsvektor der Tangente von a an Γ.

Das einfachste Beispiel einer TUY–KIRILLOV–Kurve ist eine Abtastgeometrie, die aus 2 konzentrischen, orthogonalen Kreisen mit ausreichend grossem Radius besteht. Eine alternative Charakterisierung von TUY–KIRILLOV–Kurven wird in Abschnitt 1.4.1 angegeben. Zur Berechung der Approximativen Inversen und des Rekonstruktionskerns notwendig ist die Verwendung des folgenden

Lemma 1.2.1 (Formel von GRANGEAT). Sei $\omega \in S^2$ ein Einheitsvektor und $a \in \mathbb{R}^3 \setminus \Omega^n$, δ bezeichne die Delta–Distribution. Dann gilt

$$\frac{\partial}{\partial s}\mathcal{R}f(\omega,\, a^\top\omega) = -\int_{S^2} \mathcal{D}f(a,\, \theta)\, \delta'(\theta^\top\omega)\, d\theta. \tag{1.16}$$

Geht man jetzt vom Rekonstruktionskern der RÖNTGEN-Transformation aus, beschränkt diese wieder auf $\mathsf{L}_2(Z)$, integriert partiell und setzt Lemma 1.2.1 ein, so ergibt sich für TUY–KIRILLOV–Kurven

$$\begin{aligned}
\mathcal{S}g(x) &= \frac{1}{4}\int_{S^2}\int_\Gamma\int_{S^2} \mathcal{D}f(a,\, \theta)\, \delta'(\theta^\top\omega)\, d\theta\, \Psi^0\Big(\frac{(a-x)^\top\omega}{2}\Big)\, |a'^\top\omega|\, n(\omega,\, a^\top\omega)^{-1}\, da\, d\omega \\
&= \frac{1}{4}\int_\Gamma\int_{S^2} \mathcal{D}f(a,\, \theta)\, k(x;\, a,\, \theta)\, d\theta\, da
\end{aligned}$$

mit

$$k(x;\, a,\, \theta) = \int\limits_{S^2} \Psi^0(\frac{(a-x)^\top \omega}{2})\, |a'^\top \omega|\, n(\omega,\, a^\top \omega)^{-1}\, \delta'(\theta^\top \omega)\, d\omega \ . \tag{1.17}$$

Dabei stellt $n(\omega,\, s)$ die Anzahl der Schnittpunkte mit der Ebene $x^\top \omega = s$ dar.

Kreisförmige Abtastkurve

Hier sei die Abtastkurve $\Gamma = r\, S^1$. Es ist wie schon erwähnt keine exakte Inversion möglich, daher gelangt man analog der Gewinnung von 1.17 zu einer Approximation $\widetilde{\mathcal{S}}$, die sich wie schon im letzten Abschnitt $\mathcal{S}$ aus dem Rekonstruktionskern der RÖNTGEN-Transformation ableitet. Es ergibt sich

$$\begin{aligned}
\widetilde{\mathcal{S}} g(x) &= \frac{1}{8} \int\limits_{S^2} \int\limits_{\Gamma} \int\limits_{S^2} \mathcal{D} f(a,\, \theta)\, \delta'(\theta^\top \omega)\, d\theta\, \Psi^0(\frac{(a-x)^\top \omega}{2})\, |a'^\top \omega|\, da\, d\omega \\
&= \frac{1}{8} \int\limits_{\Gamma} \int\limits_{S^2} \mathcal{D} f(a,\, \theta)\, k(x;\, a,\, \theta)\, d\theta\, da
\end{aligned}$$

mit

$$k(x;\, a,\, \theta) = \int\limits_{S^2} \Psi^0(\frac{(a-x)^\top \omega}{2})\, |a'^\top \omega|\, \delta'(\theta^\top \omega)\, d\omega \ . \tag{1.18}$$

Die Berechnung dieses Kerns ist aufwendig, kann aber durch Einsetzen einer Transformation $U\theta = e_3$, die den Richtungsvektor auf die z-Achse dreht, wesentlich vereinfacht werden. Man erhält dann:

$$\begin{aligned}
k(x;\, a,\, \theta) &= \frac{1}{2}\, (x-a)^\top \theta \int\limits_{S^2 \cap \theta^\perp} \psi^0(\frac{(a-x)^\top \omega}{2})\, |a'^\top \omega|\, d\omega - \\
&\quad - a'^\top \theta \int\limits_{S^2 \cap \theta^\perp} \Psi^0(\frac{(a-x)^\top \omega}{2})\, \mathrm{sgn}(a'^\top \omega)\, d\omega \ .
\end{aligned}$$

Dieser Kern kann, insbesondere nach weiteren Approximationen, effizient implementiert werden, da er nur für eine Quellposition und $x = 0$ berechnet werden muss.

1.3 Das GUF-Verfahren

Dieser Abschnitt beinhaltet eine ausführliche Darstellung des Verfahrens mit **G**eometrie-**U**nabhängiger Strategie der **F**ilterentwicklung, kurz GUF-Verfahren. Nach der Motivation wird im zweiten Teil das Verfahren und seine Grundlagen anschaulich beschrieben. Im dritten werden wir die verwendeten Operatoren in diskreter Form genau definieren und zu einer mathematischen Verfahrensbeschreibung gelangen. Die als Basisfunktionen dienenden Elemente und die analytische Berechnung ihrer RÖNTGEN-Transformation folgt im vierten Teil, der sich auch der Voruntersuchung im 2D-Fall widmet.

1.3.1 Motivation

Probleme bei der Filterberechnung

Wie gezeigt, ist der Rekonstruktionskern aus 1.17 und 1.18 durch Anwendung der Invarianzen effizient zu implementieren, da er im Allgemeinen nur für eine einzige Quellposition und einen Rekonstruktionspunkt berechnet werden muß. Wesentlich für die Berechnung ist allerdings die Geometrie. Sie geht in die Berechnung durch den Parameter a' und die Funktion $n(\cdot)$ ein.

Diese sind von wesentlicher Bedeutung für den Rekonstruktionskern : die Funktion n gibt die Gewichtung einer einzelnen Quellposition an und a parametrisiert die Abtastrichtung. Verwendet man spezielle Geometrien, wie die Cone–Beam–Geometrie, so lässt sich a' direkt angeben und n fixieren (z.B. als $n = 2$ bei kreisförmiger Abtastung). Für andere Geometrien kann man, wie es gebräuchlich ist, n approximieren, indem man z.B. den Mittelwert über alle n verwendet. Eine exakte Berechnung ist damit allerdings nicht zu realisieren, siehe [16, 4, 17]. Ähnliches gilt für a', das sich z.B. bei einem diskret vorliegenden Satz Projektionen über den Richtungsvektor auf die nächste Detektorposition approximieren läßt. Je nach Geometrie kann es in diesen Fällen allerdings notwendig sein, für jeden Quellpunkt einen anderen Rekonstruktionskern berechnen zu müssen, was eine erhebliche Steigerung der Rechenzeit darstellt.

Es wäre also wünschenswert, ein Rekonstruktionsverfahren anzugeben, bei dem diese Pa-

rameter keine Rolle spielen bzw. implizit in die Berechnung des Filters oder das Verfahren selbst eingehen. Aus diesem Grund wurde das GUF-Verfahren entwickelt.

Da man hier eine wesentliche Vorinformation, nämlich die Geometrieanordnung, nicht analytisch auswertet, kann man bei gut untersuchten Abtastkurven wie der Cone-Beam-Anordnung nicht davon ausgehen, daß qualitativ *bessere* Ergebnisse zu erzielen sind. Die Resultate sind bei entsprechender Datenlage bezüglich der visuellen Darstellung jedoch vergleichbar und werden erzielt, ohne dass irgendeine geometrieabhängige Vorberechnung erforderlich ist. Dies ist ein grosser Vorteil, der es auch ermöglicht, neue Geometrien schnell zu testen.

Alternative Rekonstruktionsmethoden

In diesem Kontext stellt sich die Frage, ob alternative Rekonstruktionsverfahren obigen Punkten schon genügen. Es werden als solche die Inversion mittels ϱ–filtered–Layergram und direkte Fouriermethoden in Betracht gezogen.

Beginnen wir mit der ersteren. Setzt man in der Inversionformel aus Satz 1.1.3 $\alpha = n-1$, so gelangt man zu

$$f = \frac{1}{2}(2\pi)^{1-n} I^{1-n} \mathcal{R}^{\#} g, \;\; g = \mathcal{R}f$$

Für ungerades n ergibt sich das Riesz-Potential als

$$I^{(1-n)} = (-\Delta)^{(n-1)/2}$$

mit dem Laplace-Operator Δ. Setzt man in unserem Fall $n = 3$, so kann man f aus

$$f(x) = -\frac{1}{8\pi^2}\Delta \int_{S^2} g(\theta, x \cdot \theta) d\theta \tag{1.19}$$

berechnen.

Diese Inversionsformel geht schon auf Radon zurück. Sie wird als Rekonstruktionsmethode umgesetzt, indem man f gemäß

$$f = \frac{1}{2}(2\pi)^{-2}(|\xi|^2 (\mathcal{R}^{\#} g)^{\wedge})^{\vee}, \; g = \mathcal{R}f,$$

berechnet, d.h. erst rückprojiziert und dann die FOURIER-Transformation zum Filtern einsetzt.

Die Probleme, die sich hier ergeben (siehe [31]), sind erstens die Behandlung des Nullpunktes der Fourier-Transformation der Daten, denn diese wird mit 0 gewichtet. Die Fourier–Transformation der Rückprojektion selbst besitzt dort eine Singularität, so daß das Ergebnis der Filterung nicht ohne weiteres anzugeben ist. Zweitens muß $(\mathcal{R}^{\#}g)$ zur Berechnung der FOURIER-Transformierten auf ganz $\mathbb{R}^3$ bekannt sein, wovon nicht ausgegangen werden kann.

Man kann bei realen Daten, die verrauscht und unvollständig sind, auch nicht davon ausgehen, daß eine numerische Differentiation zusammen mit den globalen Artefakten der FOURIER-Transformation immer zufriedenstellende Ergebnisse liefern.

Das Problem bei der Rekonstruktionsmethode des ρ-filtered-Layergram läßt sich also auf das Filtern nach der Rückprojektion reduzieren. Hier wäre eine Methode wünschenswert, bei der man den Filter auf bekannte Daten zurückführt und dadurch numerische Probleme vermeidet. Genau dies wird beim GUF-Verfahren angestrebt und weitgehend erreicht.

Ein weiteres Rekonstruktionsverfahren ist die sogenannte direkte Fourier-Methode, die z.B. in [29] und [1] genau untersucht wurde.

Diese Rekonstruktionsmethode beruht auf dem Fourier–Slice–Theorem $\mathcal{P}f^{\wedge}(\theta, x) = \widehat{f}(x)$, $x \in \theta^{\perp}$. Dabei werden die Daten Fourier-transformiert und an entsprechender Stelle im Spektrum eines Polargitters eingetragen. Nach Interpolation auf ein kartesisches Gitter wird dann eine inverse FOURIER-Transformation angewendet.

Problem hierbei ist, dass das Fourier–Slice–Theorem für die divergente RÖNTGEN-Transformation nicht gilt. Um es einzusetzen, müssten die Daten, die nach ihrer Quellposition und nicht nach der Richtung zu Projektionen gebündelt sind, erst umgeordnet werden (Rebinning-Methode), was im dreidimensionalen eine nicht praktikable Methode darstellt.

Es ist unter der Bedingung, daß die Quelle so weit vom Objekt entfernt ist, dass eine quasi-parallele Geometrie vorliegt, eine Approximation möglich. Dies ist jedoch wiederum eine starke Einschränkung an mögliche Geometrien.

1.3.2 Verfahrensbeschreibung

Das GUF-Verfahren basiert nicht auf der Inversionsformel (1.11), sondern verwendet den

Satz 1.3.1. Seien $f \in \mathcal{S}(\mathbb{R}^n)$ und $g \in \mathcal{S}(\widetilde{Z})$. Dann gilt

$$(\mathcal{P}^{\#} g) * f = \mathcal{P}^{\#}(g * \mathcal{P} f).$$

Beweis: Es gilt

$$\begin{aligned}(\mathcal{P}^{\#} g) * f(x) &= \int_{\mathbb{R}^n} \mathcal{P}^{\#} g(x-y) f(y) dy \\ &= \int_{\mathbb{R}^n} \int_{S^{n-1}} g(\theta, E_\theta(x-y)) d\theta f(y) dy \\ &= \int_{S^{n-1}} \int_{\mathbb{R}^n} g(\theta, E_\theta(x-y)) f(y) dy d\theta\end{aligned}$$

Substituiert man nun $y = z + t\theta$, $z \in \theta^{\perp}$, so erhält man

$$\begin{aligned}(\mathcal{P}^{\#} g) * f(x) &= \int_{S^{n-1}} \int_{\theta^{\perp}} \int_{\mathbb{R}} g(\theta, E_\theta(x-(z+t\theta))) f(z+t\theta) dt dz d\theta \\ &= \int_{S^{n-1}} \int_{\theta^{\perp}} g(\theta, E_\theta x - z)) \mathcal{P} f(\theta, z) dz d\theta \\ &= \int_{S^{n-1}} (g * \mathcal{P} f)(\theta, E_\theta x) d\theta \\ &= \mathcal{P}^{\#}(g * \mathcal{P} f)(x).\end{aligned}$$

■

Korollar 1.3.1. Bei vollständigen Daten gilt mit der Identifikation $\mathcal{D} f(a, \theta) = \mathcal{P} f(\theta, E_\theta a)$ für $supp(f) \subset \Omega$ dieser Satz auch für die *divergente* RÖNTGEN-*Transformation* .

Außerdem wird folgende Beziehung benötigt (siehe [31]).

Satz 1.3.2. Für $f \in \mathcal{S}(\mathbb{R}^n)$ ist

$$\mathcal{P}^{\#} \mathcal{P} f = 2|x|^{1-n} * f.$$

Mit $\mathcal{L}f(a,\theta) := \mathcal{D}(a,\theta) + \mathcal{D}(a,-\theta)$ gilt

$$\mathcal{L}^*\mathcal{L}f = c\frac{\Gamma(\frac{n-1}{2})}{2\pi^{n/2}\Gamma(1/2)}|x|^{1-n} * f,$$

wobei $\mathcal{L}^*$ der adjungierte Operator zu $\mathcal{L}$ ist, Γ die Gamma-Funktion und c eine Konstante.

Beweis: Eine einfache Rechnung zeigt

$$\begin{aligned}
\mathcal{P}^\#\mathcal{P}f(x) &= \int_{\mathsf{S}^{n-1}} \mathcal{P}f(\theta, E_\theta x)\mathrm{d}\theta \\
&= \int_{\mathsf{S}^{n-1}}\int_{\mathbb{R}} f(E_\theta x + t\theta)\mathrm{d}t\mathrm{d}\theta \\
&= \int_{\mathsf{S}^{n-1}}\int_{\mathbb{R}} f(x + t\theta)\mathrm{d}t\mathrm{d}\theta \\
&= 2\int_{\mathsf{S}^{n-1}}\int_0^\infty f(x + t\theta)\mathrm{d}t\mathrm{d}\theta \\
&= 2\int_{\mathbb{R}^n} f(x+y)|y|^{1-n}\mathrm{d}y, \text{ wobei } y = t\theta \text{ gesetzt wurde.}
\end{aligned}$$

Den Beweis zum zweiten Teil des Satzes findet man in [18]. ∎

Satz 1.3.2 motiviert folgende Vorgehensweise. Sei H eine Funktion mit

$$e = H * \mathcal{P}^\#\mathcal{P}e \tag{1.20}$$

für ein $e \in \mathcal{S}(\mathbb{R}^n)$. Die Idee besteht ähnlich der Approximativen Inversen darin, statt f eine geglättete Version $\widetilde{f} := f * e$ zu rekonstruieren.
Die Verfahrensbeschreibung ist wie folgt. Mit Hilfe des Faltungssatzes erhält man aus der Identität 1.20 formal den Faltungskern

$$H = (\frac{\widehat{e}}{(\mathcal{P}^\#\mathcal{P}e)^\wedge})^\vee. \tag{1.21}$$

Die Division durch $(\mathcal{P}^\#\mathcal{P}e)^\wedge$ wird in der praktischen Anwendung durch (1.24), S.43, regularisiert, um numerische Probleme zu vermeiden. Es sei $\widetilde{f} = f * e$. Dann gilt

$$\begin{aligned}\widetilde{f} = f * e &= f * (H * \mathcal{P}^{\#}\mathcal{P}e) \\ &= H * (f * \mathcal{P}^{\#}\mathcal{P}e) \\ &= H * (\mathcal{P}^{\#}\mathcal{P}e * f) \\ &= H * (\mathcal{P}^{\#}(\mathcal{P}f * \mathcal{P}e)).\end{aligned}$$

Es handelt sich also weder um ein Verfahren des Typs "gefilterte Rückprojektion", noch um den Typ "Filterung nach Rückprojektion", sondern um ein Hybrid von beidem.
Die Funktion e wird im weiteren Basisfunktion genannt. Dieser Begriff soll nun motiviert werden. Wir betrachten eine diskrete Punktmenge $\{x_i\}_i \subset \mathbb{R}^3$ und definieren $e_i := T^{x_i}e$ mit der Translation $T^{x_i}e(x) = e(x - x_i)$ und der Bedingung $\text{supp}(e_i) \cap \text{supp}(e_j) = \emptyset$ für $i \neq j$. Dann kann man $\widetilde{f}$ ausgewertet an den Punkten x_i darstellen als

$$\begin{aligned}\widetilde{f}(x_i) &= (f * e)(x_i) \\ &= \int f(y)e(y - x_i)\mathrm{d}y \\ &= \int f(y)T^{x_i}e(y)\mathrm{d}y \\ &= \int f(y)e_i(y)\mathrm{d}y \\ &= \langle f, e_i\rangle \\ &=: f_i. \end{aligned} \tag{1.22}$$

Die Rekonstruktion lässt sich schreiben als

$$\widetilde{f} = \sum_i \langle f, e_i\rangle e_i = \sum_i f_i e_i. \tag{1.23}$$

In diesem Sinn ist die Funktion e eine generierende Basisfunktion, d.h. das Rekonstruktionsergebnis wird von den Elementen $\{e_i\}_i$ aufgespannt. Durch (1.22) und (1.23), wird deutlich, dass e eine Interpolationseigenschaft hat.

Die Basisfunktion ist im einfachsten Fall eine Voxel-Basis (charakteristische Funktion eines Voxels), welche f durch Mittelung über ein Gebiet in Quaderform glättet. Weitere untersuchte Basisfunktionen umfassen die 3D-Hut-Funktion (auch "Voxel-Interpolationsbasis"), die von ihrem Maximum 1 in der Mitte linear zu 0 am Rand abfällt, die charakteristische Funktion einer Sphäre und eine sphärische Gauß-Funktion.
Verwendet man als Basisfunktion in der analytischen Beschreibung die δ-Distribution, so ergibt sich aufgrund ihrer Eigenschaften einer Inversionsformel, die der Rekonstruktion des Typs "Filterung nach Rückprojektion" entspricht. Für eine solche Rekonstruktion und Inversion siehe [31] und [42].
Notwendige Bedingung an die Basisfunktion ist, daß sowohl e, als auch $\mathcal{D}e$ (in den praktischen Untersuchungen wird die divergente RÖNTGEN-Transformation verwendet) analytisch bekannt sein müssen. Dies führt zu den Berechnungen von Abschnitt 1.3.6.
Die bisherige Verfahrensbeschreibung wurde unter der Voraussetzung *vollständiger* Daten gemacht. Mit Hilfe des Lemmas 1.1.1 kann man die Daten der divergenten RÖNTGEN-Transformation und der parallelen RÖNTGEN-Transformation ineinander überführen. Deshalb gilt die Verfahrensbeschreibung auch für $\mathcal{D}f$.
Im realen Fall liegen jedoch (abgesehen von ganz speziellen Geometrieparametern) immer nichtvollständige Daten vor. Selbst wenn die Geometrie theoretisch die Tuy–Kirillov–Bedingung erfüllt, so kann nicht für jeden Punkt des Rekonstruktionsgebietes ein Schnittpunkt der entsprechenden Ebene mit der Abtastkurve vorausgesetzt werden. Somit muß man unter Verwendung einer realen Abtastgeometrie Γ das bisherige Verfahren approximativ definieren. Es gilt

$$e \approx e_\Gamma = H * \mathcal{D}_\Gamma^* \mathcal{D}_\Gamma e.$$

Statt jetzt die Approximation e_Γ an e zu betrachten, wird ein Filterkern H_Γ über

$$H_\Gamma = \left(\frac{\widehat{e}}{(\mathcal{D}_\Gamma^* \mathcal{D}_\Gamma e)^\wedge}\right)^\vee$$

berechnet und es ergibt sich

$$\widetilde{f} = f * e \quad = \quad f * (H_\Gamma * \mathcal{D}_\Gamma^* \mathcal{D}_\Gamma e)$$

$$\begin{aligned} &= H_\Gamma * (f * \mathcal{D}_\Gamma^* \mathcal{D}_\Gamma e) \\ &= H_\Gamma * (\mathcal{D}_\Gamma^* \mathcal{D}_\Gamma e * f) \\ &\approx H_\Gamma * (\mathcal{D}_\Gamma^* (\mathcal{D}_\Gamma f * \mathcal{D}_\Gamma e)) =: \widetilde{f}_\Gamma. \end{aligned}$$

Je näher Γ an vollständigen Daten liegt, umso besser ist die Approximation von $\widetilde{f}_\Gamma$ an $\widetilde{f}$ (siehe Abschnitt 1.4.2).
Der Vorteil liegt hier darin, daß, abgesehen von der RÖNTGEN-Transformation der Basiselemente und der späteren erläuterten Regularisierung (siehe 1.3.3) keine analytische Vorberechnungen, die sonst stark von der Geometrie abhängen, in die Filterberechnung eingehen. Nachdem die Parameter des Verfahrens gewählt sind, erfolgt die Rekonstruktion ohne weitere Vorbereitung, unabhängig von der verwendeten Geometrie.

Bemerkung 1.3.1. Das GUF-Verfahren hat sich im Bezug auf die Parameterwahl als sehr robust herausgestellt: alle Rekonstruktionen mit synthetischen Daten wurden unter Verwendung der gleichen Regularisierungsparameter berechnet.

Ermöglicht wird dies dadurch, daß alle Geometrieabhängigkeiten diskret über die Berechnung der divergenten RÖNTGEN-Transformation des Basiselementes, die auf dem Detektor berechnet wird, und deren Rückprojektion in den Voxel-Kubus, die mit der Geometrie der Ausgangsdaten berechnet wird, eingehen. Man sieht somit direkt, daß es sich, um eine geometrieunabhängige Strategie der Filterentwicklung handelt.
Nachfolgend noch einige Erläuterungen zur praktischen Durchführung.
Hat man eine entsprechende Basis gewählt, so ist der Ablauf des GUF-Verfahrens folgendermaßen. Die an einer bekannten Geometrieposition gewonnenen Daten $\mathcal{D}_\Gamma f(a, \cdot)$ des Detektors werden mit der divergenten RÖNTGEN-Transformation des Basiselementes $\mathcal{D}_\Gamma e(a, \cdot)$, die analytisch an exakt der gleichen Stelle berechnet wird, an der die Daten gewonnen wurden, gefaltet. Diese modifizierten Daten $(\mathcal{D}_\Gamma f * \mathcal{D}_\Gamma e)(a, \cdot)$ werden in einen Voxel-Kubus V_d rückprojiziert, die Daten $\mathcal{D}_\Gamma e(a, \cdot)$ zusätzlich in einen anderen, V_e. So verfährt man mit allen Richtungen, so daß als Resultat die diskreten Versionen von $(\mathcal{D}_\Gamma^* (\mathcal{D}_\Gamma f * \mathcal{D}_\Gamma e))$ und $\mathcal{D}_\Gamma^* \mathcal{D}_\Gamma e$ in V_d und V_e vorliegen.

Da die exakte Funktion e bekannt ist, läßt sich der Faltungskern H gemäß Gleichung 1.21, direkt bestimmen, indem man in einen Voxel-Kubus V_{ex} eine diskrete Auswertung der Basisfunktion e rendert und den Kern mit V_{ex} und V_e berechnet.
Der letzte Schritt der Rekonstruktion besteht jetzt in der Anwendung dieses Filters auf V_d. Die letztbeschriebenen Schritte werden aus Effizienzgründen nicht über eine diskrete Faltung, sondern durch Anwendung des Faltungssatzes $f * g = (\widehat{f}\widehat{g})^\vee$ berechnet. Hierbei werden gleichzeitig zwei Regularisierungen des Kerns durchgeführt. Eine Präzisierung dieser Schritte findet man im nächsten Abschnitt.

1.3.3 GUF-Verfahren und Regularisierung

Das GUF-Verfahren stellt eine diskrete Rekonstruktionsmethode dar (in dem Sinn, daß der Filterprozeß keine Approximation an eine analytische Formel, sondern eine Adaption an die diskret vorliegenden Rekonstruktionsparameter, welche in diesem Abschnitt definiert werden, ist). Deshalb wird das Verfahren hier im konkreten Zusammenhang aller Rekonstruktionsparameter definiert. Die Wahl der Parameter entspricht der im Musterkatalog 3.1, so daß ein direkter Bezug zwischen Modell und Implementierung offensichtlich ist.
In der folgenden Verfahrensbeschreibung werden Mengen und Abbildungen betrachtet.
Die Mengen unterteilen sich in Detektoren und Voxel-Kuben. Für eine Rekonstruktion werden eine Anzahl n_{proj} Projektionen benötigt. Jede dieser Projektionen stellt in der Modellierung einen Detektor dar. Dieser ist eine Einbettung einer rechteckigen Punktmenge in den dreidimensionalen Raum, wobei die Lage und die Verkippung frei wählbar sind, was sich durch eine Einbettungsabbildung, die jedem Detektor eigen ist, ausdrückt. Zusätzlich enthält jeder Detektor den korrespondierenden Quellpunkt a, von dem aus die Strahlen bei der Datengewinnung das Objekt durchdringen. Die Detektorwerte sind durch eine Abbildung der Raumpunkte auf $\mathbb{C}$ dargestellt.
Ein Voxel-Kubus ist eine quaderförmige Punktmenge als Teilmenge des $\mathbb{R}^3$, deren Kanten standardmäßig parallel zu den Koordinatenachsen liegen und dessen Mitte sich mit dem Nullpunkt des Koordinatensystems deckt. Dieser Voxelkubus ist über die Parametrisierung

jedoch frei im Raum drehbar.

Die Abbildungen unterscheiden basisfunktionsspezifische Abbildungen, die Rückprojektion, Regularisierungen und die Fourier-Transformation. Erstere berechnen entweder die Punktauswertung der Basisfunktion für einen Voxelkubus oder die RÖNTGEN-Transformation auf einem Detektor. Die Rückprojektion bildet die Detektorwerte in einem Voxelkubus ab.

Die Regularisierungen unterteilen sich in die Operatoren $\mathsf{H}_{r_{min},r_{max},\gamma}$ und E_ϵ. Der erste schneidet durch ein radialsymmetrisches Fenster mit wählbarem Abklingverhalten, das über die Exponentialfunktion modelliert wird, die hohen Frequenzen ab. Der zweite kontrolliert die Division bei der Filterberechnung von H_Γ (siehe 1.3.2), indem nur bei Werten über einem Schwellwert Γ eine Division erfolgt. Dies ist notwendig, da das Spektrum von $\mathcal{D}_\Gamma^*\mathcal{D}_\Gamma e$ von der Geometrie abhängt und größere Bereiche betragsmäßig fast Null sein können. Eine unregularisierte Division würde zu einer starken Gewichtung von Bereichen im Spektrum führen, die keine verwertbaren Informationen enthalten.

Die Fouriertransformation wird für Detektoren und Voxelkuben definiert.

Nachdem im letzten Abschnitt der Ablauf des GUF-Verfahrens beschrieben wurde, stellt er sich bezüglich der verwendeten Mengen folgendermaßen dar: Die Daten stehen als Detektoren Δ_i zur Verfügung. Diese werden unter Verwendung der Fouriertransformation F_2 mit der auf dem Detektor $\mathcal{D}_{e_\beta}(\Delta_i)$ berechneten Basisfunktion gefaltet und mittels der Rückprojektion B auf einen Voxelkubus V_i abgebildet. Die V_i werden aufsummiert und das Resultat mit Hilfe der Fouriertransformation F_3 mit dem Voxelkubus M, der die Filtermatrix enthält, gefaltet. Den Voxelkubus M erhält man durch Rückprojektion und Summation der schon verwendeten Detektoren $\mathcal{D}_{e_\beta}(\Delta_i)$ in einen Voxelkubus V_e. Dieser bildet den Divisor in der regularisierten Division E_ϵ, wobei als Dividend die Auswertung der Basisfunktion e_β in einem Voxelkubus V_{ex} dient. Man erhält nun M dadurch, dass man auf das Resultat der letzten Operation den Regularisierungsoperator H anwendet, der die hohen Frequenzen dämpft.

Zunächst also zur Definition der Mengen:

Mathematisches Modell des Detektors

Ein Detektor ist ein Tupel $\mathbf{\Delta} := \{\Delta, i_\Delta, j_\Delta, a, \theta, \phi\}$, wobei

- $\Delta = \Delta(\Delta_{nx}, \Delta_{ny}) \subset \mathbb{N}^2$ eine Punktmenge in $\mathbb{N}^2$ mit $\Delta_{nx} \times \Delta_{ny}$ Punkten ist,
- $i_\Delta : \Delta \to \mathbb{R}^3$, $i_\Delta = i_\Delta(d_D, \bar{\theta}, \bar{\phi}, \delta_x, \delta_y)$ eine Einbettung in den $\mathbb{R}^3$ vermöge

$$i_\Delta : \{0, \ldots, \Delta_{nx} - 1\} \times \{0, \ldots, \Delta_{ny} - 1\} \to \mathbb{R}^3$$

$$(x, y) \mapsto P_\Delta := (c_x, c_y, c_z) = U_{\bar{\theta},\bar{\phi}}(d_D, \delta_x(x - \frac{\Delta_{nx}}{2})/\Delta_{nx}, \delta_y(y - \frac{\Delta_{ny}}{2})/\Delta_{ny}))$$

  mit der Drehmatrix $U_{\bar{\theta},\bar{\phi}}$ darstellt,

- $j_\Delta : \mathbb{R}^3 \to \mathbb{C}$ die Detektorwerte sind,
- a den Quellpunkt darstellt,
- $\omega(\theta, \phi)$ die Richtung Quellpunkt-Detektorzentrum in sphärischen Koordinaten darstellt.

Die Bedeutung der restlichen Parameter ist wie folgt:

- Δ_{nx}, Δ_{ny} Anzahl Detektorelemente
- δ_x, δ_y Grösse eines Detektorelementes
- d_D Distanz Isozentrum des Untersuchungsobjektes zum Zentrum des Detektors
- $\omega(\bar{\theta}, \bar{\phi})$ Richtung senkrecht zum Detektor

Aus dem Quellpunkt berechnet sich in der praktischen Untersuchung der implizite Parameter d_R, die Distanz von a zum Isozentrum des Untersuchungsobjektes. Wird auf einen Punkt Q mit $j_\Delta(Q)$ zugegriffen, für den $i_\Delta^{-1}(Q) \notin \Delta$, so seien seine Werte innerhalb des Detektorgebietes bilinear interpoliert, ausserhalb gleich Null.

Mathematisches Modell des Voxel-Kubus

Ein Voxel-Kubus ist ein Tupel $\mathbf{V} := \{V, i_V, j_V, \widetilde{\theta}, \widetilde{\phi}\}$, wobei

- $V = V(V_{nx}, V_{ny}, V_{nz}) \subset \mathbb{N}^3$ eine Punktmenge in $\mathbb{N}^3$ mit $V_{nx} \times V_{ny} \times V_{nz}$ Punkten ist,
- $i_V : V \to \mathbb{R}^3$, $i_V = i_V(d_D, \widetilde{\theta}, \widetilde{\phi}, V_x, V_y, V_z)$ eine Einbettung in den $\mathbb{R}^3$ vermöge

$$i_V : \{0 \dots V_{nx} - 1\} \times \{0 \dots V_{ny} - 1\} \times \{0 \dots V_{nz} - 1\} \to \mathbb{R}^3$$

$$\begin{aligned} (x, y, z) &\mapsto P_V := (v_x, v_y, v_z) \\ &= U_{\widetilde{\theta},\widetilde{\phi}}(V_x(x - \frac{V_{nx}}{2})/V_{nx}, V_y(y - \frac{V_{ny}}{2})/V_{ny}, V_z(z - \frac{V_{nz}}{2})/V_{nz}))) \end{aligned}$$

  mit der Drehmatrix $U_{\widetilde{\theta},\widetilde{\phi}}$ darstellt,

- $j_V : \mathbb{R}^3 \to \mathbb{C}$ die Voxelwerte sind,
- $\omega(\widetilde{\theta}, \widetilde{\phi})$ die Richtung des Voxelkubus im Raum ist.

Die Transformation $U_{\widetilde{\theta},\widetilde{\phi}}$ dreht die Standardrichtung des Voxel-Kubus, die der e_3-Achse entspricht, auf $\omega(\widetilde{\theta_v}, \widetilde{\phi_v})$ in sphärischen Koordinaten.
Aus notationstechnischen Gründen gilt $j_V(i_V(V)) = \{j_V(i_V(R)), R \in V\}$.
Der Menge V zusammen mit i_V stellt die Diskretisierung des Rekonstruktionsgebietes Ω^n dar.

Die Abbildungen sind definiert als:

Geometrie

$$\begin{aligned} \Gamma : 0 \dots n_{proj} - 1 &\rightarrow \mathbb{R}^3 \\ i &\mapsto a_i \end{aligned}$$

Die Geometrie entspricht einem Weg im $\mathbb{R}^3$ bei der der Projektionsnummer i eindeutig eine Position a_i, die durch die Datengewinnung vorgeschrieben ist, im $\mathbb{R}^3$ zugewiesen wird. Die Gesamtanzahl Projektionen ist n_{proj} Siehe dazu Abschnitt 1.4.2.

Rückprojektion

Es ist

$$\begin{aligned} B : \mathbf{\Delta} &\rightarrow \mathbf{V} \\ \{\Delta, i_\Delta, j_\Delta, a, \theta, \phi\} &\mapsto \{V, i_V, j_B, \widetilde{\theta}, \widetilde{\phi}\} \\ \text{mit } j_B(i_V(P))) &= j_\Delta \left(a + (d_R + d_D) \frac{i_V(P) - a}{||i_V(P) - a||} \right) \end{aligned}$$

die Rückprojektion einer Detektormatrix $\mathbf{\Delta}$ in einen Voxelkubus $\mathbf{V}$. Jedem Punkt P des Voxel-Kubus wird dabei ein Wert auf dem Detektor zugewiesen, der sich aus der Schnittgeraden durch $i_V(P)$ und a mit dem Detektor ergibt.

Basisfunktion

$$e_\beta \in C_0^\infty(\Omega^n) : s \mapsto e_\beta(s)$$

Basisfunktion der Grösse $\beta > 0$, d.h. entweder $\text{supp}(b) = B_{\frac{\beta}{2}}(0)$ oder $\text{supp}(b) = [-\frac{\beta}{2}, \dots, \frac{\beta}{2}]^3$, siehe Abschnitt 1.3.6. Der Parameter β wird aus notationstechnischen Gründen im folgenden ausgelassen, wenn er nicht benötigt wird. Weiterhin sei $e(\mathbf{V})$ die Basisfunktion auf einem gesamten Voxel-Kubus:

$$e(\mathbf{V}) = \mathbf{V}(V, i_V, e, \widetilde{\theta}, \widetilde{\phi}).$$

Divergente RÖNTGEN-*Transformation der Basisfunktion*

Es sei

$$\begin{aligned} \mathcal{D}_e : (a,s) \in \Gamma \times S^2 &\rightarrow \mathbb{R} \\ \mathcal{D}_e(a,s) &= \int_{(g:x=a+ts)\cap \mathrm{supp}e} e(r)dr \end{aligned}$$

die divergente RÖNTGEN-Transformation für eine Richtung s und einen Quellpunkt a bezüglich der Basisfunktion e.

Weiterhin sei

$$\begin{aligned} D_e : \mathbf{\Delta} &\mapsto \mathbf{\Delta} \\ D_e(\mathbf{\Delta}) &= \{\Delta, i_\Delta, j_D, a, \theta, \phi\} \\ &\text{mit} \quad j_D : i_\Delta(P) \mapsto \mathcal{D}_e(a,s),\ s = \frac{i_\Delta(P)-a}{||i_\Delta(P)-a||},\ P \in \Delta \end{aligned}$$

die divergente RÖNTGEN-Transformation für einen kompletten Detektor $\mathbf{\Delta}$, d.h. Integrale über alle Strahlen, die ausgehend vom Quellpunkt a ein Detektorelement treffen und den Träger der Basisfunktion schneiden.

Geometrieregularisierung

$$\mathrm{E}_\epsilon : \mathbf{V} \times \mathbf{V} \rightarrow \mathbf{V} \tag{1.24}$$

$$(\mathbf{V}_1 := \{V, i_V, j_{V1}, \widetilde{\theta}, \widetilde{\phi}\}, \mathbf{V}_2 := \{V, i_V, j_{V2}, \widetilde{\theta}, \widetilde{\phi}\}) \mapsto \mathbf{V}_3 := \{V, i_V, j_{\epsilon,V}, \widetilde{\theta}, \widetilde{\phi}\})$$

mit

$$j_{\epsilon,V}(i_V(P)) := \begin{cases} j_{V1}(i_V(P))/j_{V2}(i_V(P)) & \text{für } |(j_{V2}(i_V(P))| \geq \epsilon \\ 0 & \text{sonst} \end{cases}$$

Mit E_ϵ wird im Filter der Einfluss der Geometrie regularisiert, denn bei der Filterberechnung werden gerade die Stellen, welche keine Informationen enthalten, stark gewichtet, was zusätzliche Artefakte einführt. Dem wird mit Hilfe dieser Regularisierung entgegengewirkt. Der Parameter ϵ definiert dabei den Schwellwert. In $\mathbf{V}_1$, $\mathbf{V}_2$ und $\mathbf{V}_3$ stimmen V, i_V, $\widetilde{\theta}$ und $\widetilde{\phi}$ überein.

Frequenzregularisierung

$$\begin{aligned} \mathsf{H}_{r_{min},r_{max},\gamma} : \mathbf{V} &\rightarrow \mathbf{V} \\ \mathbf{V} = \{V, i_V, j_V, \widetilde{\theta}, \widetilde{\phi}\} &\mapsto \mathsf{H}_{r_{min},r_{max},\gamma}(\mathbf{V}) := \{V, i_V, j_{h,V}, \widetilde{\theta}, \widetilde{\phi}\} \\ \text{wobei } j_{h,V}(i_V(P)) &:= \exp(-\gamma(t * \mathrm{H}(t))^2) j_V(i_V(P)) \\ \text{mit } t &:= \frac{(||i_V(P)|| - r_{min})}{r_{max} - r_{min}} \end{aligned}$$

Die Funktion H dämpft die hohen Frequenz mittels eines radialsymmetrischen GAUSS-Fensters mit frei definierbarer Breite (von r_{min} bis r_{max}) und der Stärke des Abklingverhaltens γ. Die Funktion H stellt die Heavyside-Funktion dar.

FOURIER-*Transformationen*

Die 2-dimensionale Fouriertransformation F_2 auf dem Detektor $\mathbf{\Delta}$ ist definiert als

$$\begin{aligned} F_2 : \mathbf{\Delta} &\rightarrow \mathbf{\Delta} \\ \{\Delta, i_\Delta, j_\Delta, a, \theta, \phi\} &\mapsto \{\Delta, i_\Delta, j_{F,\Delta}, a, \theta, \phi\} \\ \text{mit } j_{F,\Delta}(i_\Delta(i_1, i_2)) &= \sum_{j_1=0}^{\Delta_{nx}-1} \sum_{j_2=0}^{\Delta_{ny}-1} j_\Delta(i_\Delta(j_1, j_2)) \omega_1^{-i_1 j_1} \omega_2^{-i_2 j_2} \end{aligned}$$

wobei $\omega_1 = e^{2\pi\imath/\Delta_{nx}}$ und $\omega_2 = e^{2\pi\imath/\Delta_{ny}}$.

Die inverse FOURIER-Transformation wird mit F_2^{-1} bezeichnet und berechnet sich als

$$\begin{aligned} F_2^{-1} : \mathbf{\Delta} &\rightarrow \mathbf{\Delta} \\ \{\Delta, i_\Delta, j_\Delta, a, \theta, \phi\} &\mapsto \{\Delta, i_\Delta, j_{-F,\Delta}, a, \theta, \phi\} \\ \text{mit } j_{-F,\Delta}(i_\Delta(i_1, i_2)) &= \frac{1}{\Delta_{nx}\Delta_{ny}} \sum_{j_1=0}^{\Delta_{nx}-1} \sum_{j_2=0}^{\Delta_{ny}-1} j_\Delta(i_\Delta(j_1, j_2)) \omega_1^{i_1 j_1} \omega_2^{i_2 j_2} \end{aligned}$$

Die 3-dimensionale FOURIER-Transformation F_3 und ihre Inverse F_3^{-1} sind analog definiert und operieren lediglich auf einem Voxelkubus $\mathbf{V}$ statt einem Detektor $\mathbf{\Delta}$.

Mit diesen Operatoren definiert sich das GUF-Verfahren als

$$\mathbf{V} = F_3^{-1}(MF_3(\sum_i B(F_2^{-1}(F_2(\mathbf{\Delta}_i)F_2(D_{e_\beta}(\mathbf{\Delta}_i)))$$

mit

$$M = \mathsf{H}_{r_{min},r_{max},\gamma}(\mathrm{E}_\epsilon(F_3(b_\beta), F_3(\sum_i B(D_{e_\beta}(\mathbf{\Delta}_i)))))$$

bezüglich der Daten $\{\mathbf{\Delta}_i\}$ und den Regularisierungsparametern β, r_{min}, r_{max}, γ, ϵ

.

Von den Regularisierungsparametern ist nur β vor der Rückprojektion, die den zeitaufwendigsten Schritt der Rekonstruktion darstellt, zu wählen. Das optimale β kann gut abgeschätzt werden, denn es stellt die Grösse der Basisfunktion dar und ist nur von der Detektor- und Voxeldiskretisierung abhängig, die vorher bekannt sind. Die anderen Parameter, die den Filterungsprozess steuern, können nachträglich variiert werden, um das Rekonstruktionsergebnis zu optimieren. Dieser Schritt ist nicht zeitaufwendig und liegt bei geschickter Programmierung im Sekundenbereich.

Bemerkung 1.3.2. Die diskrete Formulierung des GUF-Verfahrens macht deutlich, dass eine geschickte Implementierung unbedingt erforderlich ist. Das Klassenframework TORNE (Kapitel 2) wurde u.a. geschaffen, um dieser Notwendigkeit gerecht zu werden.

1.3.4 Voruntersuchung: der 2D-Fall

Das Verfahren wurde an Rekonstruktionen der Dimension $n = 2$ getestet. Der prinzipielle Vorgehensweise ist exakt die gleiche. Hier ist der Vorteil, dass bei einer Beschränkung auf parallele Geometrie, wie es der Fall war, verschiedene Symmetrieeigenschaften der Basiselemente e ausgenutzt werden konnten, die bei der analytischen Berechnung der RÖNTGEN-Transformation von Vorteil sind. Siehe dazu und dem verwendeten Phantom [42].

1.3.5 Basisfunktionen im 2D-Fall

Die Basisfunktionen, die im 2D-Fall eingesetzt wurden, sind Funktionen, deren Träger einfache 2-dimensionale Objekte sind. Notwendige Voraussetzung ist hierbei, dass die Röntgen- oder RADON-Transformation (welche sich in 2D nur durch ihre Parametrisierung unterscheiden) analytisch berechnet werden kann. Durch die Parametrisierung der Transformation über Winkel und Abstand ist im Allgemeinen die Berechnung einfacher als im 3D-Fall und es können Symmetrieeigenschaften ausgenutzt werden.
Bezüglich der Notation wird die RADON-Transformation $R(\theta, s)$ mit $R(\omega(\theta), s)$ identifiziert.

1.3.5.1 Charakteristische Funktion des Kreises

Die Basisfunktion ist

$$\mathrm{e}(x) := \begin{cases} 1 & : \quad x \in \mathrm{B}^2_R(0) \subset \mathbb{R}^2 \\ 0 & : \quad sonst \end{cases},$$

wobei $R = \beta/2$ der Radius ist.
Ihre Radon-Transformation ist einfach zu berechnen, da die Basisfunktion radialsymmetrisch ist und somit die Richtung keine Rolle spielt:

$$\mathrm{Re}(\theta, s) := \begin{cases} 2\sqrt{R^2 - s^2} & : \quad |s| \leq R \\ 0 & : \quad sonst \end{cases}$$

1.3.5.2 Charakteristische Funktion des Quadrates

Diese Basisfunktion hat die Gestalt

$$\mathrm{e}(x) := \begin{cases} 1 & : \quad x \in \mathrm{Q}^2(0) := \{y \in \mathbb{R}^2 \mid ||y||_\infty \leq 1\} \\ 0 & : \quad \text{sonst} \end{cases},$$

Um dieses Element auf die Grösse $d = \beta/2$ zu dilatieren, wird die Dilatationsinvarianz der RADON-Transformation ausgenutzt.
Bei der Berechnung der RADON-Transformation kann man die Symmetrie

$$Re(\theta, s) = Re(\theta + t\frac{\pi}{2}, |s|), \ t \in \mathbb{Z}$$

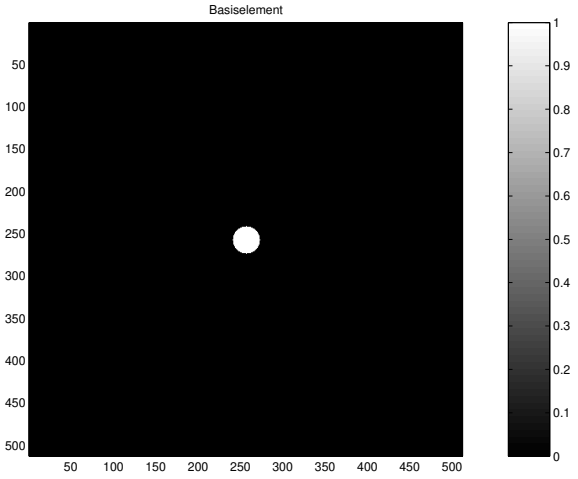

Abbildung 1.2: Radialbasis

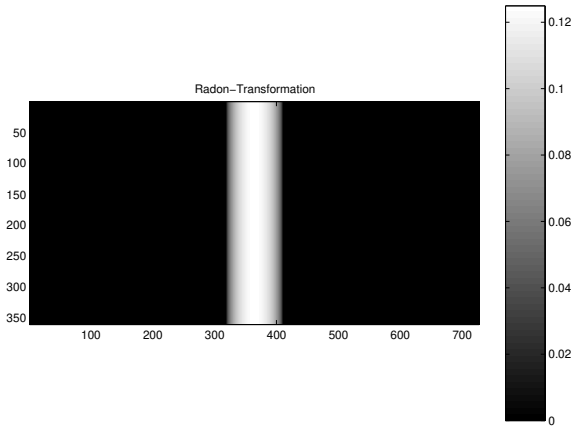

Abbildung 1.3: Radontransformation der Radialbasis

ausnutzen.

Nach Bestimmung der Schnittpunkte

$$\begin{aligned} x_l &= \frac{s}{\cos(\theta)} + \tan(\theta); \\ x_u &= \frac{s}{\cos(\theta)} - \tan(\theta); \\ y_l &= \frac{s}{\sin(\theta)} + \cot(\theta); \\ y_u &= \frac{s}{\sin(\theta)} - \cot(\theta); \end{aligned}$$

ergibt sich die RADON-Transformation durch Fallunterscheidung:

$$\begin{aligned} |x_l| \leq 1 \wedge |x_u| \leq 1 : \; Re(\theta, s) &= 2/\cos(\theta) \\ |x_l| > 1 \wedge |x_u| \leq 1 : \; Re(\theta, s) &= \sqrt{(1 - x_u)^2 + (1 - y_u)^2} \\ |y_l| \leq 1 \wedge |y_u| \leq 1 : \; Re(\theta, s) &= 2/\sin(\theta) \\ \text{sonst: } Re(\theta, s) &= 0 \end{aligned}$$

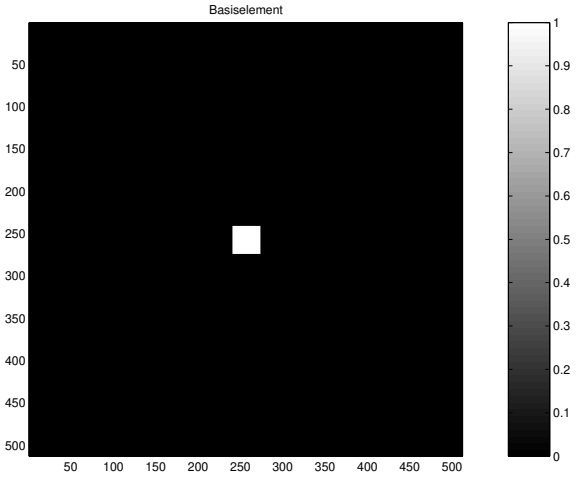

Abbildung 1.4: Quadratbasis

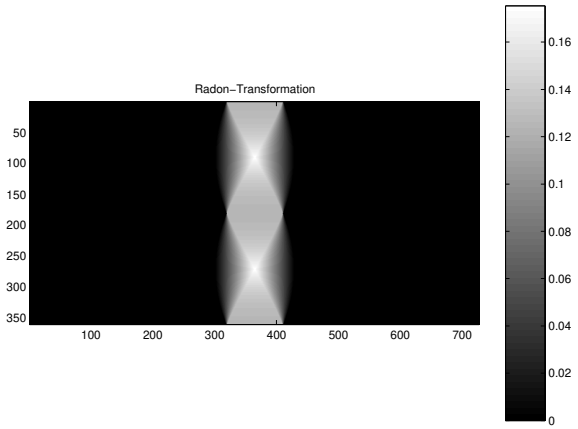

Abbildung 1.5: Radontransformation der Quadratbasis

1.3.5.3 Radiale Gauss-Funktion in 2D

Die Funktion ist

$$\mathrm{e}(x) := \begin{cases} \exp(-d\langle x,x\rangle) & : \quad x \in \mathrm{B}_R^2(0) \\ 0 & : \quad sonst \end{cases}$$

mit dem Parameter d zur Steuerung des Abklingverhaltens, kann man wieder die bestehende Radialsymmetrie ausnutzen und gelangt zu

$$\begin{aligned} \mathrm{Re}(\omega, s) &= \int_{\mathbb{R}} e(s\omega + t\omega^{\perp})\mathrm{d}t \\ &= \exp(-ds^2)\int_{\mathbb{R}} \exp(-dt^2)\mathrm{d}t \\ &= \sqrt{\pi}\exp(-ds^2) \end{aligned}$$

für $|s| \leq R$.

1.3.5.4 Pyramidale Pixel-Basis

Mit

$$\mathrm{e}(x) := \begin{cases} (1-|x_0|)(1-|x_1|) & : \quad x = (x_0, x_1) \in \mathrm{Q}^2(0) \\ 0 & : \quad \text{sonst} \end{cases}$$

der Pyramiden- oder Hut-Funktion wird, im Gegensatz zur charakteristischen Funktion des Quadrates, die eine Nearest-Neighbour-Approximation darstellt, linear interpoliert.

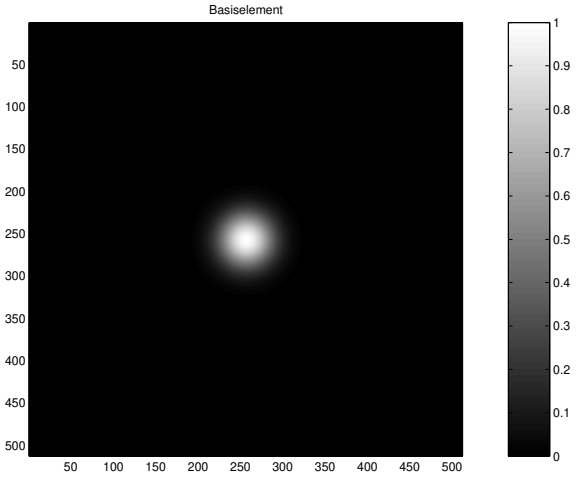

Abbildung 1.6: Gaussbasis

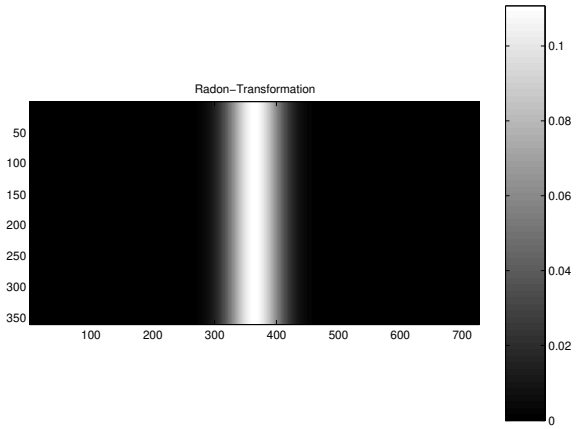

Abbildung 1.7: Radontransformation der Gaussbasis

Die hier vorliegende Symmetrie ist die gleiche wie in 1.3.5.2. Eine Grössenänderung erfolgt wieder über die Dilatationsinvarianz. Die RADON-Transformation gewinnt man über eine Fallunterscheidung:
Für $\sin(\theta) = 0$ ist

$$Re(\theta, s) = 1 - s$$

Ansonsten gilt mit

$$\begin{aligned} c_\theta &:= \cos(\theta) \\ s_\theta &:= \sin(\theta) \end{aligned}$$

und den Schnittpunkten

$$\begin{aligned} y_0 &= \frac{s}{s_\theta} \\ y_1 &= \frac{s - c_\theta}{s_\theta} \end{aligned}$$

für die RADON-Transformation :

$$(s > -s_\theta + c_\theta) \wedge (s > s_\theta) \wedge (y_1 < 0) :$$

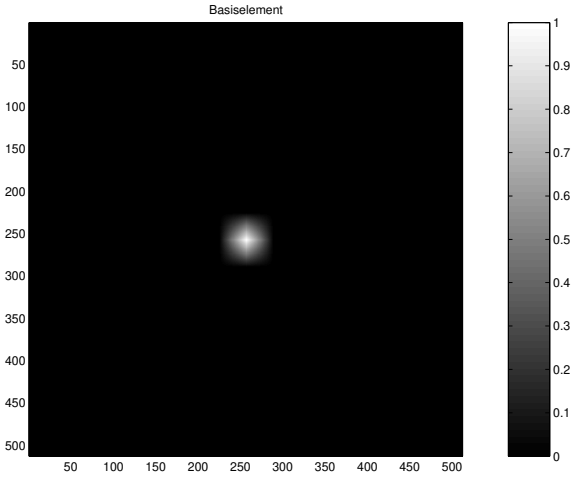

Abbildung 1.8: Pyramidalbasis

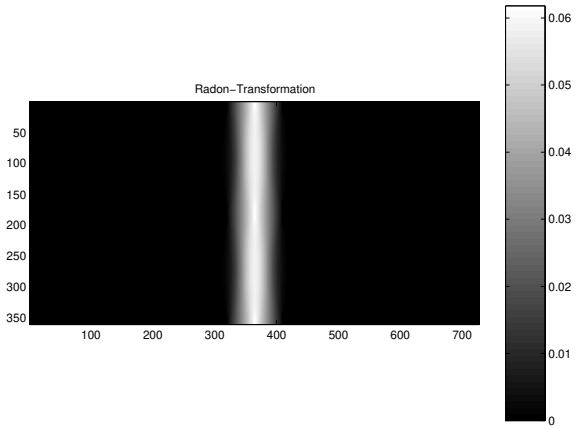

Abbildung 1.9: Radontransformation der Pyramidalbasis

$$\mathrm{Re}(\theta,s) = \frac{c_\theta(1-y_1)-\frac{1}{2}c_\theta(1+y_1^2)+s(\frac{1}{2}y_1-1)+\frac{1}{2}(s_\theta+y_1c_\theta)+\frac{1}{2}s+\frac{1}{3}(y_1^2(c_\theta+\frac{1}{2}s)-s_\theta)}{{c_\theta}^2}$$

$$(s > -s_\theta + c_\theta) \wedge (s > s_\theta) \wedge (y_1 \geq 0):$$

$$\mathrm{Re}(\theta,s) = \frac{c_\theta(1-y_1)-\frac{1}{2}c_\theta(1-y_1^2)+s(\frac{1}{2}y_1-1)+\frac{1}{2}(s_\theta+y_1c_\theta)+\frac{1}{2}s-\frac{(y_1^2(c_\theta+\frac{1}{2}s)+s_\theta)}{3}}{{c_\theta}^2}$$

$$(s > -s_\theta + c_\theta) \wedge (s \leq s_\theta) \wedge (y_1 < 0):$$

$$\mathrm{Re}(\theta,s) = \frac{c_\theta(1-y_1)-\frac{1}{2}c_\theta(1+y_1^2)+s(1-y_0+\frac{1}{2}y_1)+\frac{1}{2}(y_1c_\theta-s_\theta)+\frac{s_\theta+y_0^2s+y_1^2(c_\theta+\frac{1}{2}s)}{3}-\frac{s}{2}}{{c_\theta}^2}$$

$$(s > -s_\theta + c_\theta) \wedge (s \leq s_\theta) \wedge (y_1 \geq 0):$$

$$\mathrm{Re}(\theta,s) = \frac{c_\theta(1-y_1)-\frac{1}{2}c_\theta(1-y_1^2)+s(1-y_0+\frac{1}{2}y_1)+\frac{1}{2}(y_1c_\theta-s_\theta)+\frac{(s_\theta+y_0^2s-y_1^2(c_\theta+\frac{1}{2}s))}{3}-\frac{s}{2}}{{c_\theta}^2}$$

$$(s \leq -s_\theta + c_\theta) \wedge (s > s_\theta):$$

$$\mathrm{Re}(\theta,s) = \frac{2c_\theta - c_\theta - 2s + s}{{c_\theta}^2}$$

$$(s \leq -s_\theta + c_\theta) \wedge (s \leq s_\theta):$$

$$\mathrm{Re}(\theta,s) = \frac{2c_\theta - c_\theta - s_\theta - \frac{s^2}{s_\theta} + \frac{2s_\theta + sy_0^2}{3}}{{c_\theta}^2}$$

1.3.6 Basisfunktionen im 3D-Fall

Wie im 2D-Fall ist die Basisfunktion e gekennzeichnet durch ihre geometrische Form, den Träger, und die Funktion, die auf dem Träger lebt. Parametrisiert man den Röntgenstrahl als $x = a + ts$ mit Quellpunkt $a = (a_x, a_y, a_z)^T$ und Richtungsvektor $s = (s_x, s_y, s_z)^T$, $s \in \mathsf{S}^2$, so besteht die prinzipielle Vorgehensweise darin, die beiden Schnittpunkte $s_0 = a + t_0 s$ und $s_1 = a + t_1 s$ zu finden und das Integral

$$\mathcal{D}e(a,s) = \int_{s_0}^{s_1} e(x)\mathrm{d}x = \int_{t_0}^{t_1} e(a+ts)\mathrm{d}t$$

zu bestimmen.

1.3.6.1 Charakteristische Funktion der Kugel

Die Basisfunktion ist hier bei gegebenem Radius $R = \beta/2$

$$\mathrm{e}(x) := \chi_{\mathrm{B}_R(0)}(x) = \begin{cases} 1 & : \quad x \in \mathrm{B}_R^3(0) \\ 0 & : \quad sonst \end{cases}$$

Die Schnittpunkte berechnen sich als

$$\begin{aligned} t_0 &= R^2 - < a,a > + < a,s >^2 - < a,s > \\ &= R^2 - a_x^2 - a_y^2 - a_z^2 + (a_x s_x + a_y s_y + a_z s_z)^2 - (a_x s_x + a_y s_y + a_z s_z) \\ t_1 &= -R^2 + < a,a > - < a,s >^2 - < a,s > \\ &= -R^2 + a_x^2 + a_y^2 + a_z^2 - (a_x s_x + a_y s_y + a_z s_z)^2 - (a_x s_x + a_y s_y + a_z s_z) \end{aligned}$$

Die Röntgen-Transformation ist

$$\mathcal{D}e(a,s) = \int_{t_0}^{t_1} e(a+ts)\,\mathrm{d}t = t_1 - t_0$$

1.3.6.2 Charakteristische Funktion des Würfels

Sei die Basisfunktion ein Würfel um den Nullpunkt mit Abstand d, also Kantenlänge $2d = \beta$:

$$\mathrm{e}(x) := \begin{cases} 1 & : \quad x \in \mathrm{Q}_d^3(0) := \{y \in \mathbb{R}^3 |\ ||y||_\infty \leq d\} \\ 0 & : \quad \text{sonst} \end{cases}$$

Man erhält die Schnittpunkte dadurch, dass man die sechs Seiten zu Ebenen im Raum fortsetzt, deren Schnittpunkte mit dem Röntgenstrahl berechnet und ihre Lage innerhalb der Würfelgrenzen verifiziert. Der Röntgenstrahl sei wie in 1.3.6 parametrisiert.
Seien E_0 bis E_5 die Würfelebenen, parametrisiert durch die Normalform

$$E_l : x^T n_l - d_l = 0, \quad l = 1..6$$

mit den Normalenvektoren gleich den Einheitsvektoren

$$n_i = e_i, \quad n_{i+3} = e_i, \quad i = 1, 2, 3,$$

und den Abständen

$$d_i = d, \quad d_{i+3} = -d, \quad i = 1, 2, 3.$$

Die Ebenenschnittpunkte erhält man als

$$t_i = \frac{d_l - [a]_i}{[s]_i}.$$

Die Punkte t_0 und t_1 sind dann diejenigen, für die $|t_i| = d$ gilt. Es sei $t_0 < t_1$.
Die Röntgen-Transformation ist wiederum

$$\mathcal{D}e(a, s) = \int_{t_0}^{t_1} e(a + ts)\,\mathrm{d}t = t_1 - t_0$$

1.3.6.3 Radiale Gauss-Basis in 3D

Mit dem Parameter d für das Mass des Abklingens ist die Basisfunktion hier

$$\mathrm{e}(x) := \begin{cases} \exp(-d\langle x, x\rangle) & : \quad x \in \mathrm{B}_R^3(0) \\ 0 & : \quad sonst \end{cases}$$

Sind die Schnittpunkte wie in Abschnitt 1.3.6.1 berechnet und über t_0 und t_1 parametrisiert, so erhält man

$$\begin{aligned}
\mathcal{D}e(a,s) &= \int_{t_0}^{t_1} e^{-d(<a+ts,a+ts>)}\mathrm{d}t \\
&= \int_{t_0}^{t_1} e^{-d((a_x+ts_x)^2+(a_y+ts_y)^2+(a_z+ts_z)^2)}\mathrm{d}t \\
&= \int_{t_0}^{t_1} e^{-d(t^2+(2\,a_x\,s_x+2\,a_y\,s_y+2\,a_z\,s_z)\,t+{a_x}^2+{a_y}^2+{a_z}^2)}\mathrm{d}t \\
&= \frac{\sqrt{\pi}}{2\sqrt{d}}\, e^{(d\,(-{a_z}^2-{a_x}^2-{a_y}^2+{a_x}^2\,{s_x}^2+2\,a_x\,s_x\,a_y\,s_y+2\,a_x\,s_x\,a_z\,s_z+{a_y}^2\,{s_y}^2+2\,a_y\,s_y\,a_z\,s_z+{a_z}^2\,{s_z}^2))} \\
&\quad (\mathrm{erf}(\sqrt{d}\,(t_1+\,a_x\,s_x+\,a_y\,s_y+\,a_z\,s_z))-\mathrm{erf}(\sqrt{d}\,(t_0+\,a_x\,s_x+\,a_y\,s_y+\,a_z\,s_z)))
\end{aligned}$$

mit

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$$

Die Error-Funktion erf wird, da sie ein Spezialfall der unvollständigen Gamma-Funktion ist, wie letztere über die Auswertung einer Reihendarstellung implementiert. Es gilt der Zusammenhang

$$\mathrm{erf(x)} = P(\frac{1}{2}, x^2)$$

mit

$$P(t,x) = \frac{\gamma(t,x)}{\Gamma(t)} = \Gamma(t)^{-1}\int_0^x s^{t-1}e^{-s}\mathrm{ds}$$

wobei

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\mathrm{d}t$$

ist.

1.3.6.4 Pyramidale Voxel-Basis

Basisfunktion ist hier

$$\mathrm{e}(x) := \begin{cases} \prod_{i=0}^{2}(1-|x_i|) & : \quad x=(x_0,x_1,x_2)\in \mathrm{Q}_1^3(0) \\ 0 & : \quad \text{sonst} \end{cases}$$

Die Berechnung der Schnittpunkte erfolgt analog zu Abschnitt 1.3.6.2. Die Grösse des Basiselements wird über die Dilatationsinvarianz der RÖNTGEN-Transformation variiert oder

dadurch, dass man die Schnittgerade auf ein Koordinatensystem umskaliert, in dem das Basiselement die Grösse 1 besitzt. Die Auswertung des Integrals gestaltet sich hier jedoch etwas komplexer, da der Integrand mehrere Betrags-Funktionen enthält. Die geschlossene Form ergibt sich als

$$
\begin{aligned}
\mathcal{D}e(a,s) &= \int_{t_0}^{t_1} e(a+ts)\,\mathrm{d}t \\
&= \int_{t_0}^{t_1} (1-|a_x+ts_x|)(1-|a_y+ts_y|)(1-|a_z+ts_z|)dt \\
&= \frac{1}{4}\,{t_0}^4\,\mathrm{sgn}(t_0\,s_x+a_x)\,\mathrm{sgn}(t_0\,s_y+a_y)\,\mathrm{sgn}(t_0\,s_z+a_z)\,s_x\,s_y\,s_z \\
&\quad -\frac{1}{4}\,{t_1}^4\,\mathrm{sgn}(t_1\,s_y+a_y)\,\mathrm{sgn}(t_1\,s_z+a_z)\,\mathrm{sgn}(t_1\,s_x+a_x)\,s_x\,s_y\,s_z \\
&\quad +\frac{1}{3}\,{t_0}^3\,\mathrm{sgn}(t_0\,s_x+a_x)\,\mathrm{sgn}(t_0\,s_y+a_y)\,\mathrm{sgn}(t_0\,s_z+a_z)\,a_x\,s_y\,s_z \\
&\quad +\frac{1}{3}\,{t_0}^3\,\mathrm{sgn}(t_0\,s_x+a_x)\,\mathrm{sgn}(t_0\,s_y+a_y)\,\mathrm{sgn}(t_0\,s_z+a_z)\,s_x\,a_y\,s_z \\
&\quad +\frac{1}{3}\,{t_0}^3\,\mathrm{sgn}(t_0\,s_x+a_x)\,\mathrm{sgn}(t_0\,s_y+a_y)\,\mathrm{sgn}(t_0\,s_z+a_z)\,s_x\,s_y\,a_z \\
&\quad -\frac{1}{3}\,{t_1}^3\,\mathrm{sgn}(t_1\,s_y+a_y)\,\mathrm{sgn}(t_1\,s_z+a_z)\,\mathrm{sgn}(t_1\,s_x+a_x)\,a_x\,s_y\,s_z \\
&\quad -\frac{1}{3}\,{t_1}^3\,\mathrm{sgn}(t_1\,s_y+a_y)\,\mathrm{sgn}(t_1\,s_z+a_z)\,\mathrm{sgn}(t_1\,s_x+a_x)\,s_x\,a_y\,s_z \\
&\quad -\frac{1}{3}\,{t_1}^3\,\mathrm{sgn}(t_1\,s_y+a_y)\,\mathrm{sgn}(t_1\,s_z+a_z)\,\mathrm{sgn}(t_1\,s_x+a_x)\,s_x\,s_y\,a_z \\
&\quad -\frac{1}{3}\,{t_0}^3\,\mathrm{sgn}(t_0\,s_x+a_x)\,\mathrm{sgn}(t_0\,s_y+a_y)\,s_x\,s_y \\
&\quad -\frac{1}{3}\,{t_0}^3\,\mathrm{sgn}(t_0\,s_x+a_x)\,\mathrm{sgn}(t_0\,s_z+a_z)\,s_x\,s_z \\
&\quad -\frac{1}{3}\,{t_0}^3\,\mathrm{sgn}(t_0\,s_y+a_y)\,\mathrm{sgn}(t_0\,s_z+a_z)\,s_y\,s_z \\
&\quad +\frac{1}{2}\,{t_0}^2\,\mathrm{sgn}(t_0\,s_x+a_x)\,\mathrm{sgn}(t_0\,s_y+a_y)\,\mathrm{sgn}(t_0\,s_z+a_z)\,a_x\,a_y\,s_z \\
&\quad +\frac{1}{2}\,{t_0}^2\,\mathrm{sgn}(t_0\,s_x+a_x)\,\mathrm{sgn}(t_0\,s_y+a_y)\,\mathrm{sgn}(t_0\,s_z+a_z)\,a_x\,s_y\,a_z \\
&\quad +\frac{1}{2}\,{t_0}^2\,\mathrm{sgn}(t_0\,s_x+a_x)\,\mathrm{sgn}(t_0\,s_y+a_y)\,\mathrm{sgn}(t_0\,s_z+a_z)\,s_x\,a_y\,a_z \\
&\quad +\frac{1}{3}\,{t_1}^3\,\mathrm{sgn}(t_1\,s_y+a_y)\,\mathrm{sgn}(t_1\,s_z+a_z)\,s_y\,s_z \\
&\quad +\frac{1}{3}\,{t_1}^3\,\mathrm{sgn}(t_1\,s_y+a_y)\,\mathrm{sgn}(t_1\,s_x+a_x)\,s_x\,s_y
\end{aligned}
$$

$$
\begin{aligned}
&+ \frac{1}{3}\,{t_1}^3\,\mathrm{sgn}(t_1\,s_z + a_z)\,\mathrm{sgn}(t_1\,s_x + a_x)\,s_x\,s_z \\
&- \frac{1}{2}\,{t_1}^2\,\mathrm{sgn}(t_1\,s_y + a_y)\,\mathrm{sgn}(t_1\,s_z + a_z)\,\mathrm{sgn}(t_1\,s_x + a_x)\,a_x\,a_y\,s_z \\
&- \frac{1}{2}\,{t_1}^2\,\mathrm{sgn}(t_1\,s_y + a_y)\,\mathrm{sgn}(t_1\,s_z + a_z)\,\mathrm{sgn}(t_1\,s_x + a_x)\,a_x\,s_y\,a_z \\
&- \frac{1}{2}\,{t_1}^2\,\mathrm{sgn}(t_1\,s_y + a_y)\,\mathrm{sgn}(t_1\,s_z + a_z)\,\mathrm{sgn}(t_1\,s_x + a_x)\,s_x\,a_y\,a_z \\
&- \frac{1}{2}\,{t_0}^2\,\mathrm{sgn}(t_0\,s_x + a_x)\,\mathrm{sgn}(t_0\,s_y + a_y)\,a_x\,s_y \\
&- \frac{1}{2}\,{t_0}^2\,\mathrm{sgn}(t_0\,s_x + a_x)\,\mathrm{sgn}(t_0\,s_y + a_y)\,s_x\,a_y \\
&- \frac{1}{2}\,{t_0}^2\,\mathrm{sgn}(t_0\,s_x + a_x)\,\mathrm{sgn}(t_0\,s_z + a_z)\,a_x\,s_z \\
&- \frac{1}{2}\,{t_0}^2\,\mathrm{sgn}(t_0\,s_x + a_x)\,\mathrm{sgn}(t_0\,s_z + a_z)\,s_x\,a_z \\
&- \frac{1}{2}\,{t_0}^2\,\mathrm{sgn}(t_0\,s_y + a_y)\,\mathrm{sgn}(t_0\,s_z + a_z)\,a_y\,s_z \\
&- \frac{1}{2}\,{t_0}^2\,\mathrm{sgn}(t_0\,s_y + a_y)\,\mathrm{sgn}(t_0\,s_z + a_z)\,s_y\,a_z \\
&+ t_0\,\mathrm{sgn}(t_0\,s_x + a_x)\,\mathrm{sgn}(t_0\,s_y + a_y)\,\mathrm{sgn}(t_0\,s_z + a_z)\,a_x\,a_y\,a_z \\
&+ \frac{1}{2}\,{t_1}^2\,\mathrm{sgn}(t_1\,s_y + a_y)\,\mathrm{sgn}(t_1\,s_z + a_z)\,a_y\,s_z \\
&+ \frac{1}{2}\,{t_1}^2\,\mathrm{sgn}(t_1\,s_y + a_y)\,\mathrm{sgn}(t_1\,s_z + a_z)\,s_y\,a_z \\
&+ \frac{1}{2}\,{t_1}^2\,\mathrm{sgn}(t_1\,s_y + a_y)\,\mathrm{sgn}(t_1\,s_x + a_x)\,a_x\,s_y \\
&+ \frac{1}{2}\,{t_1}^2\,\mathrm{sgn}(t_1\,s_y + a_y)\,\mathrm{sgn}(t_1\,s_x + a_x)\,s_x\,a_y \\
&+ \frac{1}{2}\,{t_1}^2\,\mathrm{sgn}(t_1\,s_z + a_z)\,\mathrm{sgn}(t_1\,s_x + a_x)\,a_x\,s_z \\
&+ \frac{1}{2}\,{t_1}^2\,\mathrm{sgn}(t_1\,s_z + a_z)\,\mathrm{sgn}(t_1\,s_x + a_x)\,s_x\,a_z \\
&- t_1\,\mathrm{sgn}(t_1\,s_y + a_y)\,\mathrm{sgn}(t_1\,s_z + a_z)\,\mathrm{sgn}(t_1\,s_x + a_x)\,a_x\,a_y\,a_z \\
&+ \frac{1}{2}\,{t_0}^2\,\mathrm{sgn}(t_0\,s_x + a_x)\,s_x + \frac{1}{2}\,{t_0}^2\,\mathrm{sgn}(t_0\,s_y + a_y)\,s_y \\
&+ \frac{1}{2}\,{t_0}^2\,\mathrm{sgn}(t_0\,s_z + a_z)\,s_z - t_0\,\mathrm{sgn}(t_0\,s_x + a_x)\,\mathrm{sgn}(t_0\,s_y + a_y)\,a_x\,a_y \\
&- t_0\,\mathrm{sgn}(t_0\,s_x + a_x)\,\mathrm{sgn}(t_0\,s_z + a_z)\,a_x\,a_z \\
&- t_0\,\mathrm{sgn}(t_0\,s_y + a_y)\,\mathrm{sgn}(t_0\,s_z + a_z)\,a_y\,a_z - \frac{1}{2}\,{t_1}^2\,\mathrm{sgn}(t_1\,s_y + a_y)\,s_y \\
&- \frac{1}{2}\,{t_1}^2\,\mathrm{sgn}(t_1\,s_z + a_z)\,s_z - \frac{1}{2}\,{t_1}^2\,\mathrm{sgn}(t_1\,s_x + a_x)\,s_x
\end{aligned}
$$

$$
\begin{aligned}
&+ t_1 \operatorname{sgn}(t_1\, s_y + a_y) \operatorname{sgn}(t_1\, s_z + a_z)\, a_y\, a_z \\
&+ t_1 \operatorname{sgn}(t_1\, s_y + a_y) \operatorname{sgn}(t_1\, s_x + a_x)\, a_x\, a_y \\
&+ t_1 \operatorname{sgn}(t_1\, s_z + a_z) \operatorname{sgn}(t_1\, s_x + a_x)\, a_x\, a_z + t_0 \operatorname{sgn}(t_0\, s_x + a_x)\, a_x \\
&+ t_0 \operatorname{sgn}(t_0\, s_y + a_y)\ a_y + t_0 \operatorname{sgn}(t_0\, s_z + a_z)\, a_z \\
&- t_1 \operatorname{sgn}(t_1\, s_y + a_y)\ a_y - t_1 \operatorname{sgn}(t_1\, s_z + a_z)\, a_z \\
&- t_1 \operatorname{sgn}(t_1\, s_x + a_x)\ a_x - t_0 + t_1
\end{aligned}
$$

mit der Definition

$$
\operatorname{sgn}(x) := \begin{cases} -1 & : \quad x < 0, \\ 0 & : \quad x = 0, \\ 1 & : \quad x > 0. \end{cases} \tag{1.25}
$$

Diese Darstellung des Integrals ist theoretisch leicht zu implementieren, hat aber den Nachteil, dass immer alle Formelteile ausgewertet werden. Ausserdem ist aufgrund numerischer Ungenauigkeiten in der Funktion 1.25 $x = 0 \Leftrightarrow |x| < EPS$ zu betrachten, was zu Fehlern in der Berechnung führen kann.

Aus diesen Gründen ist die Darstellung mittels Fallunterscheidung besser geeignet. Kürzt man die Vorzeichen (Signa) der einzelnen Ausdrücke des Integranden entsprechend ab, so ergibt sich

Signa $[+\,+\,+]$:

$$
\begin{aligned}
&\int_{t_0}^{t_1} (1 + a_x + s_x\, t)\,(1 + a_y + s_y\, t)\,(1 + a_z + s_z\, t) dt = \\
&\quad \frac{1}{4}\, s_x\, s_y\, s_z\, ({t_1}^4 - {t_0}^4) + \frac{1}{3}\,(((1 + a_x)\, s_y + s_x\,(1 + a_y))\, s_z + s_x\, s_y\,(1 + a_z))\,({t_1}^3 - {t_0}^3) \\
&\quad + \frac{1}{2}\,((1 + a_x)\,(1 + a_y)\, s_z + ((1 + a_x)\, s_y + s_x\,(1 + a_y))\,(1 + a_z))\,({t_1}^2 - {t_0}^2) \\
&\quad + (1 + a_x)\,(1 + a_y)\,(1 + a_z)\,(t_1 - t_0)
\end{aligned}
$$

Signa $[+\,+\,-]$:

$$
\begin{aligned}
&\int_{t_0}^{t_1} (1 + a_x + s_x\, t)\,(1 + a_y + s_y\, t)\,(1 - a_z - s_z\, t) dt = \\
&\quad - \frac{1}{4}\, s_x\, s_y\, s_z\, ({t_1}^4 - {t_0}^4) + \frac{1}{3}\,(-((1 + a_x)\, s_y + s_x\,(1 + a_y))\, s_z + s_x\, s_y\,(1 - a_z))\,({t_1}^3 - {t_0}^3)
\end{aligned}
$$

$$+\frac{1}{2}\left(-(1+a_x)\,(1+a_y)\,s_z+((1+a_x)\,s_y+s_x\,(1+a_y))\,(1-a_z)\right)\left({t_1}^2-{t_0}^2\right)$$
$$+(1+a_x)\,(1+a_y)\,(1-a_z)\,(t_1-t_0)$$

Signa $[+-+]$:

$$\int_{t_0}^{t_1}(1+a_x+s_x\,t)\,(1-a_y-s_y\,t)\,(1+a_z+s_z\,t)dt=$$
$$-\frac{1}{4}\,s_x\,s_y\,s_z\,\left({t_1}^4-{t_0}^4\right)+\frac{1}{3}\left(\left(-(1+a_x)\,s_y+s_x\,(1-a_y)\right)s_z-s_x\,s_y\,(1+a_z)\right)\left({t_1}^3-{t_0}^3\right)$$
$$+\frac{1}{2}\left((1+a_x)\,(1-a_y)\,s_z+(-(1+a_x)\,s_y+s_x\,(1-a_y))\,(1+a_z)\right)\left({t_1}^2-{t_0}^2\right)$$
$$+(1+a_x)\,(1-a_y)\,(1+a_z)\,(t_1-t_0)$$

Signa $[+--]$:

$$\int_{t_0}^{t_1}(1+a_x+s_x\,t)\,(1-a_y-s_y\,t)\,(1-a_z-s_z\,t)dt=$$
$$\frac{1}{4}\,s_x\,s_y\,s_z\,\left({t_1}^4-{t_0}^4\right)+\frac{1}{3}\left(-\left(-(1+a_x)\,s_y+s_x\,(1-a_y)\right)s_z-s_x\,s_y\,(1-a_z)\right)\left({t_1}^3-{t_0}^3\right)$$
$$+\frac{1}{2}\left(-(1+a_x)\,(1-a_y)\,s_z+(-(1+a_x)\,s_y+s_x\,(1-a_y))\,(1-a_z)\right)\left({t_1}^2-{t_0}^2\right)$$
$$+(1+a_x)\,(1-a_y)\,(1-a_z)\,(t_1-t_0)$$

Signa $[-++]$:

$$\int_{t_0}^{t_1}(1-a_x-s_x\,t)\,(1+a_y+s_y\,t)\,(1+a_z+s_z\,t)dt=$$
$$-\frac{1}{4}\,s_x\,s_y\,s_z\,\left({t_1}^4-{t_0}^4\right)+\frac{1}{3}\left(\left((1-a_x)\,s_y-s_x\,(1+a_y)\right)s_z-s_x\,s_y\,(1+a_z)\right)\left({t_1}^3-{t_0}^3\right)$$
$$+\frac{1}{2}\left((1-a_x)\,(1+a_y)\,s_z+((1-a_x)\,s_y-s_x\,(1+a_y))\,(1+a_z)\right)\left({t_1}^2-{t_0}^2\right)$$
$$+(1-a_x)\,(1+a_y)\,(1+a_z)\,(t_1-t_0)$$

Signa $[-+-]$:

$$\int_{t_0}^{t_1}(1-a_x-s_x\,t)\,(1+a_y+s_y\,t)\,(1-a_z-s_z\,t)dt=$$
$$\frac{1}{4}\,s_x\,s_y\,s_z\,\left({t_1}^4-{t_0}^4\right)+\frac{1}{3}\left(-\left((1-a_x)\,s_y-s_x\,(1+a_y)\right)s_z-s_x\,s_y\,(1-a_z)\right)\left({t_1}^3-{t_0}^3\right)$$
$$+\frac{1}{2}\left(-(1-a_x)\,(1+a_y)\,s_z+((1-a_x)\,s_y-s_x\,(1+a_y))\,(1-a_z)\right)\left({t_1}^2-{t_0}^2\right)$$
$$+(1-a_x)\,(1+a_y)\,(1-a_z)\,(t_1-t_0)$$

Signa $[--+]$:

$$\int_{t_0}^{t_1} (1 - a_x - s_x\, t)\,(1 - a_y - s_y\, t)\,(1 + a_z + s_z\, t)dt =$$
$$\frac{1}{4}\, s_x\, s_y\, s_z\, ({t_1}^4 - {t_0}^4) + \frac{1}{3}\,((-(1 - a_x)\, s_y - s_x\,(1 - a_y))\, s_z + s_x\, s_y\,(1 + a_z))\,({t_1}^3 - {t_0}^3)$$
$$+ \frac{1}{2}\,((1 - a_x)\,(1 - a_y)\, s_z + (-(1 - a_x)\, s_y - s_x\,(1 - a_y))\,(1 + a_z))\,({t_1}^2 - {t_0}^2)$$
$$+ (1 - a_x)\,(1 - a_y)\,(1 + a_z)\,(t_1 - t_0)$$

Signa $[---]$:

$$\int_{t_0}^{t_1} (1 - a_x - s_x\, t)\,(1 - a_y - s_y\, t)\,(1 - a_z - s_z\, t)dt =$$
$$-\frac{1}{4}\, s_x\, s_y\, s_z\, ({t_1}^4 - {t_0}^4) + \frac{1}{3}\,(-(-(1 - a_x)\, s_y - s_x\,(1 - a_y))\, s_z + s_x\, s_y\,(1 - a_z))\,({t_1}^3 - {t_0}^3)$$
$$+ \frac{1}{2}\,(-(1 - a_x)\,(1 - a_y)\, s_z + (-(1 - a_x)\, s_y - s_x\,(1 - a_y))\,(1 - a_z))\,({t_1}^2 - {t_0}^2)$$
$$+ (1 - a_x)\,(1 - a_y)\,(1 - a_z)\,(t_1 - t_0)$$

Hier muss allerdings beachtet werden, dass sich das Vorzeichen auf dem Integrationsweg ändern kann, so dass ausser der reinen Berechnung der Schnittpunkte für jede Basiselementauswertung die Gleichung

$$a + t * s = 0$$

komponentenweise gelöst und die Integrationsgrenzen entsprechend gewählt werden müssen.

Bemerkung 1.3.3. Es hat sich in der Praxis gezeigt, dass die numerische Berechnung trotz Fallunterscheidung mit einer relativ hohen EPS-Schranke (oder ϵ-Schranke) durchgeführt werden muss, da die Vorzeichen bei kleinen Werten aufgrund der Rechengenauigkeit zum Oszillieren neigen. In der Implementierung in TORNE wurde $EPS = 10^{-8}$ gewählt, was bezüglich der Rekonstruktionsgüte kein Problem darstellt.

1.4 Rekonstruktionen

1.4.1 Voruntersuchung: Der 2D-Fall

Die Voruntersuchungen im 2D-Fall wurden aus Praktikabilitätsgründen durchgeführt. Es wurde hier die einfachste Strahlanordnung, die parallele verwendet, und als Phantom das sogenannte MPH-Phantom (siehe [42]). Als Testgeometrien wurde die Standardanordnung, also ein Vollkreis um das Objekt, gewählt, um die Wirkungsweise der verschiedenen Basiselemente im optimalen Fall zu testen.
Die Ergebnisse sind den Abbildungen 1.10 bis 1.13 zu entnehmen. Aus diesen ist ersichtlich, dass die Güte der Rekonstruktionen bei den Basisfunktionen mit quadratischem Träger deutlich besser sind. Diese sind auch gegenüber Grössenveränderungen des Trägers stabiler, d.h. das Intervall der Trägergrösse, in dem die Rekonstruktion visuell akzeptabel ist, ist grösser. Dies liegt u.a. daran, dass Funktionen mit quadratischem Träger für die Bildschirmdarstellung, die über Pixel geschieht, besser geeignet sind.

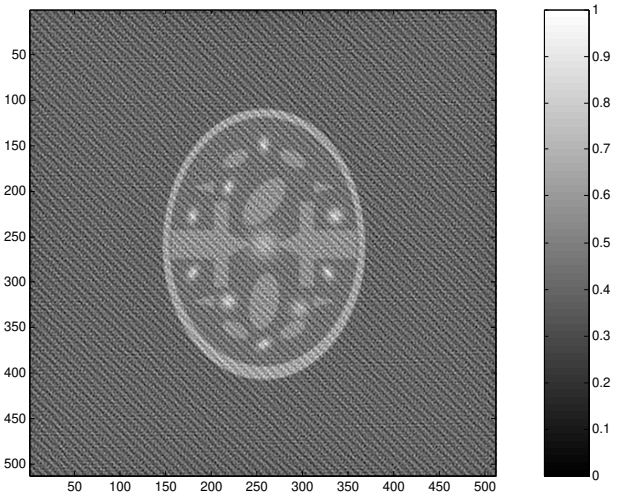

Abbildung 1.10: Rekonstruktion mit char. Funktion des Kreises

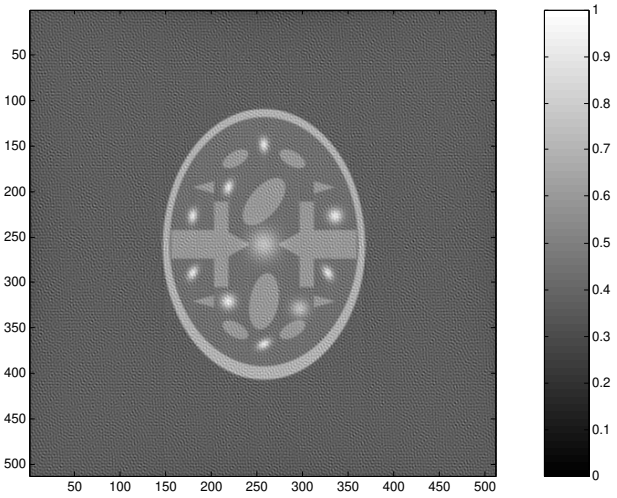

Abbildung 1.11: Rekonstruktion mit radialer Gauss-Basis

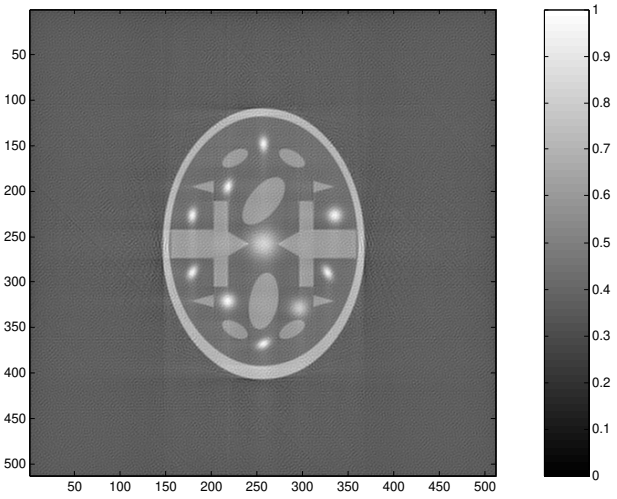

Abbildung 1.12: Rekonstruktion mit char. Funktion des Quadrates

1.4.2 Der 3D-Fall

In den nachfolgenden Rekonstruktionsbeispielen wird die Geometrie immer als Weg $\Gamma : [0,1] \to \mathbb{R}^3$ betrachtet. In den meisten Fällen gilt $\Gamma : [0,1] \to d_R \mathsf{S}^2$, der Weg liegt als Teilmenge dilatierter Standardrichtungen vor. Für die Parametrisierung von S^2 gilt wie

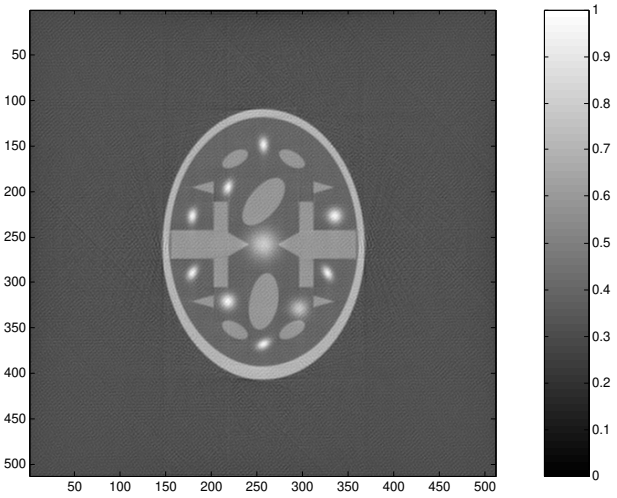

Abbildung 1.13: Rekonstruktion mit pyramidaler Pixel-Basis

üblich bei der Koordinatentransformation in sphärische Koordinaten

$$K_1 : [0, 2\pi] \times [0, \pi] \to \mathbb{R}^3$$

bzw.

$$K_2 : [0, \pi] \times [0, 2\pi] \to \mathbb{R}^3$$

$$K_{1,2} : (\theta, \phi) \mapsto \begin{pmatrix} \sin(\theta)\cos(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\theta) \end{pmatrix}$$

Zylinderkoordinaten werden parametrisiert durch

$$K_3 : [0, 2\pi] \times [-1, 1] \to \mathbb{R}^3$$

$$K_3 : (\phi, z) \mapsto \begin{pmatrix} \cos(\phi) \\ \sin(\phi) \\ zh_z \end{pmatrix}$$

mit der Zylinderhöhe $\frac{h_z}{2}$.

Das verwendete Phantom ist in Abbildung 1.14 exakt dargestellt. Es entspricht einem, in modifizierter Form weit verbreiteten Standardphantom, das eine Betrachtung von Auflösung und Dichtesprüngen ermöglicht. Zu den Details der Ellipsoide siehe Anhang A. Die Nummerierung bezieht sich auf den Anhang.
Der Aufbau dieses Phantoms ist wie folgt: die Hauptuntersuchungsebene ist die x/y-Ebene, wo die Ellipsoide so angeordnet sind, daß ihre Mittelpunkte auf drei konzentrischen Kreisen liegen. Der äußere Kreis (Nr. 9-16) umfaßt acht Ellipsoide, von den jeweils die Hälfte nach unten und nach oben mit inkrementeller Betragsdifferenz vom Durchschnittswert der Umgebung abweichen. Dies ermöglicht eine Auflösungsuntersuchung von Dichtesprüngen. Der mittlere Ring (Nr. 17-33) enthält Ellipsoide mit abnehmender Größe und konstantem Dichteunterschied, welche eine Untersuchung der Größenauflösung ermöglichen.
Der innere Ring (Nr. 34-40) enthält Ellipsoide mit Dämpfungskoeffizienten realer Materialien (Dentin, Zahnschmelz, Knochen, Luft, etc.), wobei das dichteste, Dentin, durch seinen hohen Funktionswert in der Rekonstruktion gleichzeitig zur Untersuchung von Artefakten dienen kann.
Die Ellipsoide Nr. 1-7 liegen auf der z-Achse zentriert, um die Rekonstruktion der y/z- bzw. x/z-Ebene zu untersuchen.
Die Umgebung wird über Ellipsoid Nr. 8 parametrisiert und hat den Dämpfungskoeffizienten 1.
Die untersuchten Geometrien an sich sind teilweise praxisrelevant (Orthogonale Kreise, Spirale, Cone-Beam), teilweise haben sie den Charakter eines Benchmarktests (Sinusoid, sphärische Spirale), um die Geometrieunabhängigkeit des Verfahrens zu testen.
Als Geometrieparameter wurden meistens folgende Kennzahlen verwendet:

- $d_R = d_D = 11$
- $\Delta_{nx} = \Delta_{ny} = 771$
- $V_{nx} = V_{ny} = V_{nz} = 200$
- $N_{proj} = 720$

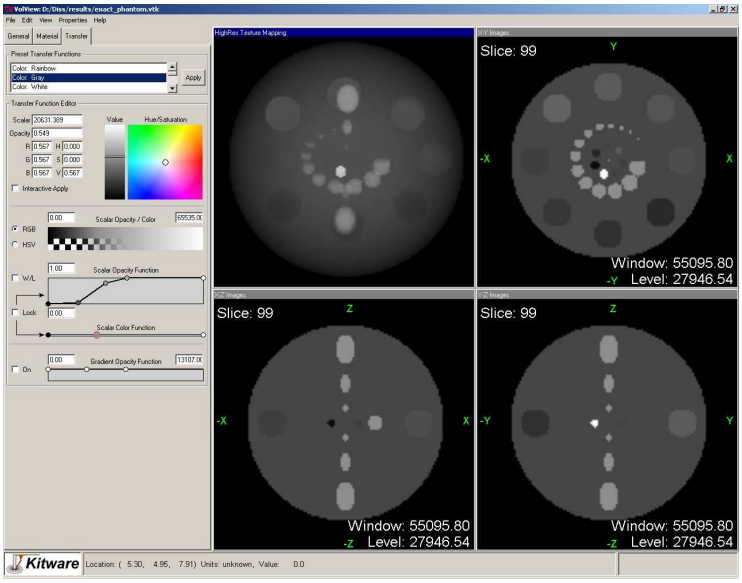

Abbildung 1.14: Das verwendete Phantom

- Voxel-Kubus-Größe: 10, Detektorgröße: 15
- Basisfunktion: pyramidale Voxel-Basis, Basisgröße $\beta = 0.2$

Unterschiede liegen nur bei der Rekonstruktion realer Daten vor, bei denen das Gesamtsystem skaliert wurde, und bei der Tomosynthese, bei der die $d_R = 55$ gewählt wurde.

1.4.2.1 Approximation an vollständige Daten

"Monte-Carlo"-Geometrie

Die "Monte-Carlo"-Geometrie (siehe Abbildung 1.4.2.1) stellt als einzige, hier betrachtete, eine Approximation an vollständige Daten dar. Es wird zufällig ein $\phi \in [0, 2\pi]$ und $\theta \in [0, \pi]$ gewählt und $\Gamma : (\theta, \phi) \mapsto K_2(\theta, \phi)$. Für $n_{proj} \to \infty$ approximiert diese Geometrie vollständige Daten. Da der Weg als solcher nicht parametrisierbar ist, kann hier bezüglich der Formel (1.17) weder eine analytische Aussage über $n(\cdot)$ noch über a' getroffen werden.

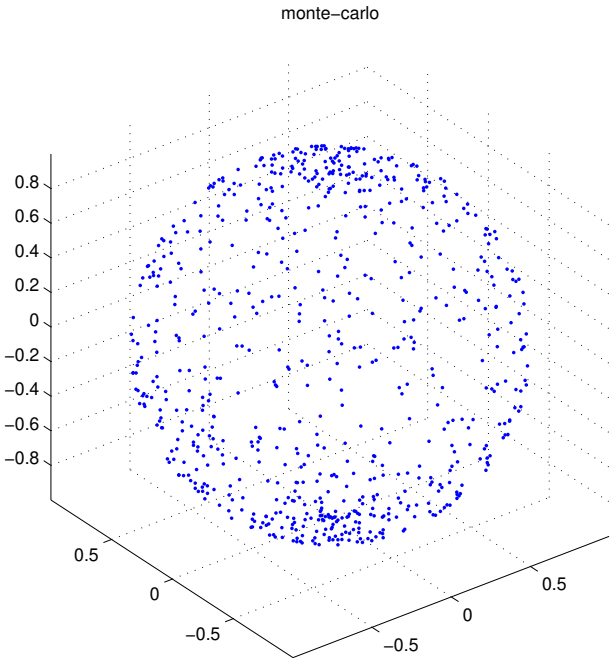

Abbildung 1.15: Monte-Carlo-Geometrie

Die Rekonstruktion mit dieser Geometrie (Abb. 1.16) liefert gute Resultate. Die leichten Artefakte in der Nähe der $z = 0$-Schicht rühren von einer Punkteverteilung her, die diese Ebene nicht gewichtet. Wie sich auch bei anderen Geometrien zeigen wird, ist die Anzahl der Quellpunkte in der x/y-Ebene entscheidend für deren Auflösung. Man kann die Güte der Rekonstruktionen bei der Monte-Carlo-Geometrie durch reines Erhöhen der Projektionsanzahl problemlos steigern.

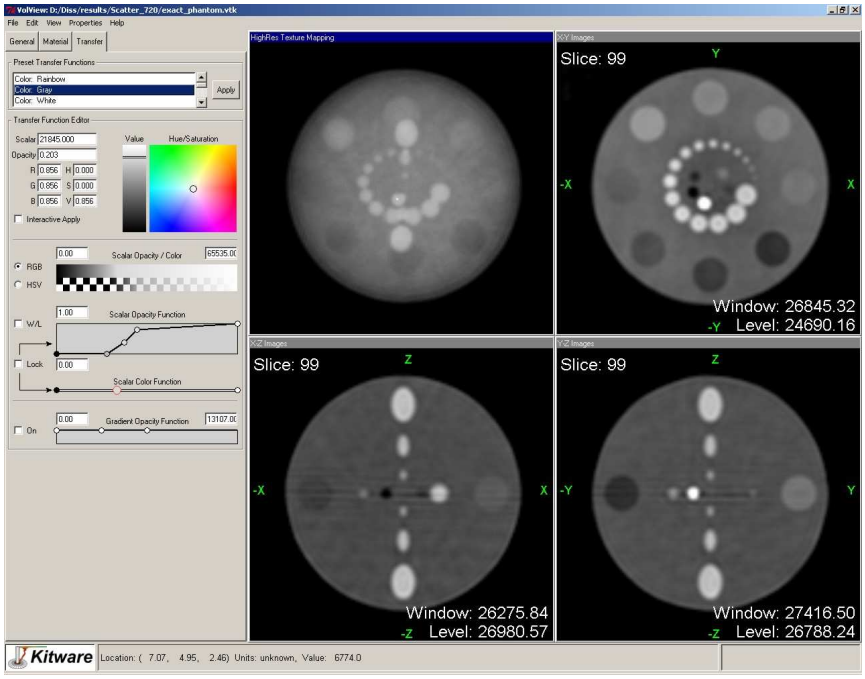

Abbildung 1.16: Rekonstruktion für die Monte-Carlo-Geometrie

1.4.2.2 Tuy-Kirillov-Geometrien

Ein Definition von TUY–KIRILLOV–Kurven wurde schon in Definition 1.2.1 gegeben. Wir wollen nun eine alternative Charakterisierung dieser Kurven definieren. Wir beginnen mit Kurven, die auf der Kugeloberfläche $\partial B_R(0)$ liegen. O.B.d.A. sei die 0 das Zentrum.
Sei also $\Gamma : [0,1] \mapsto \partial B_R(0) \subset \mathbb{R}^3$ ein stetiger Weg im $\mathbb{R}^3$. Wir betrachten nun das, im Bezug auf sein Volumen, grösste Kugelsegment s von $B_R(0)$, das Γ nicht schneidet, bzw. die entsprechende Halbkugel, falls das Segment grösser als diese ist. Das Segment s wird durch den Schnitt von $B_R(0)$ mit einer Ebene E erzeugt. Diese Ebene definieren wir als $E : x \cdot \theta - d = 0$, d.h. sie hat den Abstand $d < R$ vom Mittelpunkt. Dann gilt:

Lemma 1.4.1. Mit den obigen Bezeichnungen ist eine Kurve Γ genau dann eine TUY–KIRILLOV–Kurve bezüglich des Rekonstruktionsgebietes $\Omega_r := B_r(0)$, wenn für den Radius r gilt: $r < d < R$ mit $0 \leq d < R$. Liegt Γ nicht auf einer Kugeloberfläche, so wähle man für die Charakterisierung $B_R(0)$ mit $R = \text{diam}(\Gamma)$, also eine Kugel mit dem Radius der am weitesten entfernten Punkte. Mit dieser gilt dann obige Charakterisierung.

Siehe dazu Diagramm 1.17. Der Beweis ist aus geometrischen Gründen offensichtlich.
Mit Hilfe dieses Lemmas lässt sich für jede Kurve das maximale Rekonstruktionsgebiet angeben, in dem eine exakte Inversion möglich ist.

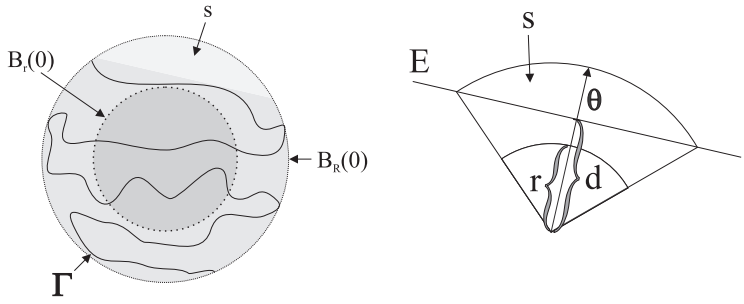

Abbildung 1.17: Tuy-Kirillov-Charakterisierung

Beispiel 1.4.1. Wählt man als Rekonstruktionsgebiet $B_1(0)$, so erfüllt die Standard-Anordnung zweier orthogonaler Kreise aus 1.4.2.2 die Bedingung 1.4.1 genau dann, wenn

die Kreise auf $B_R(0)$ mit $R > \sqrt{2}$ liegen.

Die folgenden Geometrien erfüllen alle die TUY–KIRILLOV–Bedingung laut Lemma 1.4.1 bezüglich der gewählten Rekonstruktionsparameter und ermöglichen somit theoretisch, d.h. bei unendlich vielen Daten, eine exakte Inversion.

Orthogonale Kreise

Diese Geometrie ist die Standardanordnung für die TUY–KIRILLOV–-Bedingung. Der Weg sei über

$$\Gamma = \Gamma_1 \cup \Gamma_2$$

$$\Gamma_1(t) := d_R K_1(\pi/2, \pi t)$$

$$\Gamma_2(t) := d_R K_1(2\pi t, \bar{\phi})$$

parametrisiert. Dabei kann $\bar{\phi} \in [0, \pi]$ frei gewählt werden, ohne die Orthogonalitätsbedingung zu verletzen. In den Rekonstruktionen wurde $\bar{\phi} = 0$ gesetzt.

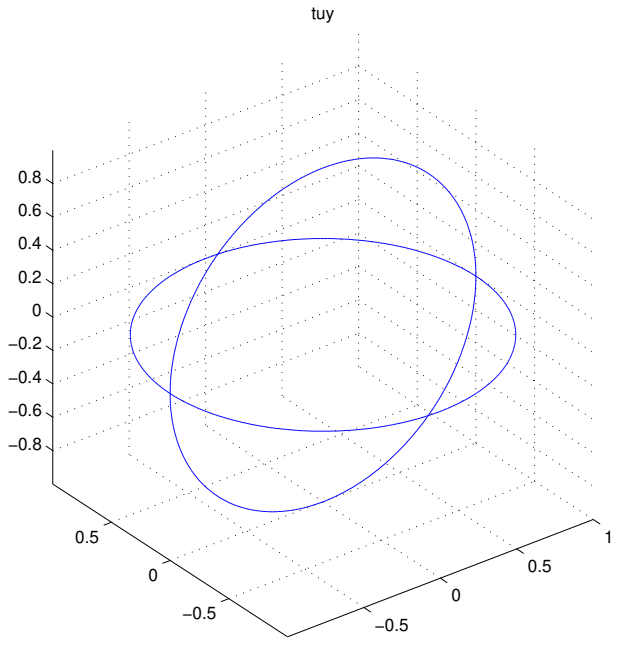

Abbildung 1.18: TK-Kreis-Geometrie

Die Rekonstruktion (Abb. 1.19) liefert sehr gute Resultate, da die Punkte in Γ in Hinsicht auf Lage im Raum und Verteilung auf dem Weg optimal gewählt sind.

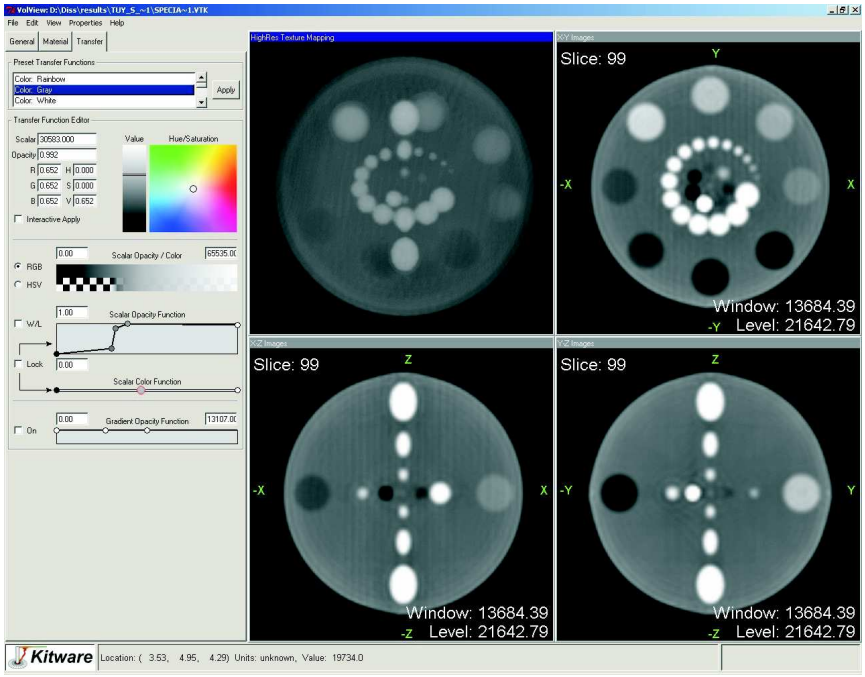

Abbildung 1.19: Rekonstruktion für die Tuy–Kirillov–Kreis–Geometrie

Sinusoid

Diese Anordnung ist die Projektion zweier oder mehrerer Sinus-Schwingungen auf eine Kugeloberfläche, wobei die Endpunkte der Schwingungen auf den Polen der Kugel liegen. Stellt p (p gerade) die Anzahl der Schwingungen dar, so ergibt sich

$$\Gamma = \bigcup_{i=1}^{p/2} (\Gamma_{d,i} \cup \Gamma_{u,i})$$

$$\Gamma_{d,i}(t) := K_2(\theta(t), \phi(t,i))$$

mit

$$\theta(t) = t\pi,\ \phi(t,i) = i^{-1}(\pi(\sin(2\theta) + 1) + 2\pi t)$$

bzw.

$$\Gamma_{u,i}(t) := K_2(\theta(t), \phi(t,i))$$

mit

$$\theta(t) = (1-t)\pi,\ \phi(t,i) = i^{-1}(\pi(\sin(2\theta) + 1) + 2\pi t).$$

Die Indizes d und u weisen dabei auf die Richtungen der Schwingung "down" und "up" hin.

Diese Geometrie liefert gute Rekonstruktionen (siehe Abb. 1.21). Artefakte liegen im Aussenraum und in der Form des Umgebungsellipsoids vor. Grund für beides ist die Lage der Punkte: sie decken das Rekonstruktionsgebiet abgesehen von den etwas grösseren Lücken, wo der Abstand der Sinusschwingungen maximal ist, relativ gut ab und die Punkte sind einigermassen gleichverteilt auf der Abtastkurve. Überraschend ist die Güte der Rekonstruktion in der x/y-Ebene, in der sich relativ wenige Abtastpunkte befinden, was jedoch durch die große Anzahl Quellpunkte am Schnittpunkt der Sinus-Schwingungen erklärbar ist, die dort einen Ausschnitt zweier orthogonaler Kreise approximieren (siehe dazu auch die Analyse der Spiral-Geometrie).

Verbessern kann man die Rekonstruktionen, indem man die Anzahl der Schwingungen und Projektionen erhöht.

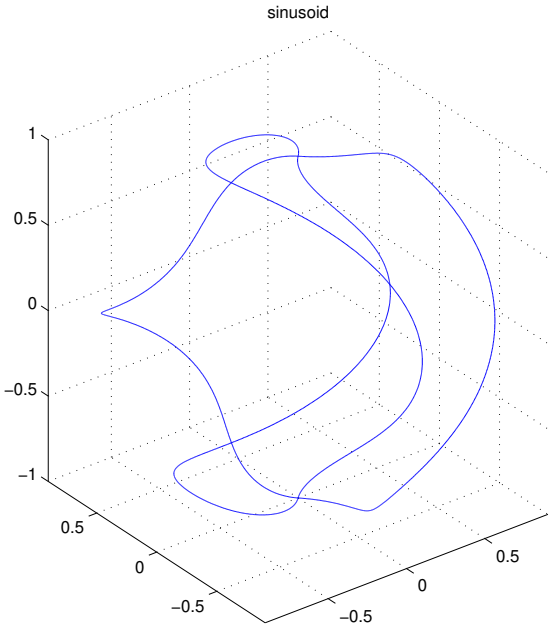

Abbildung 1.20: Sinusoid-Geometrie

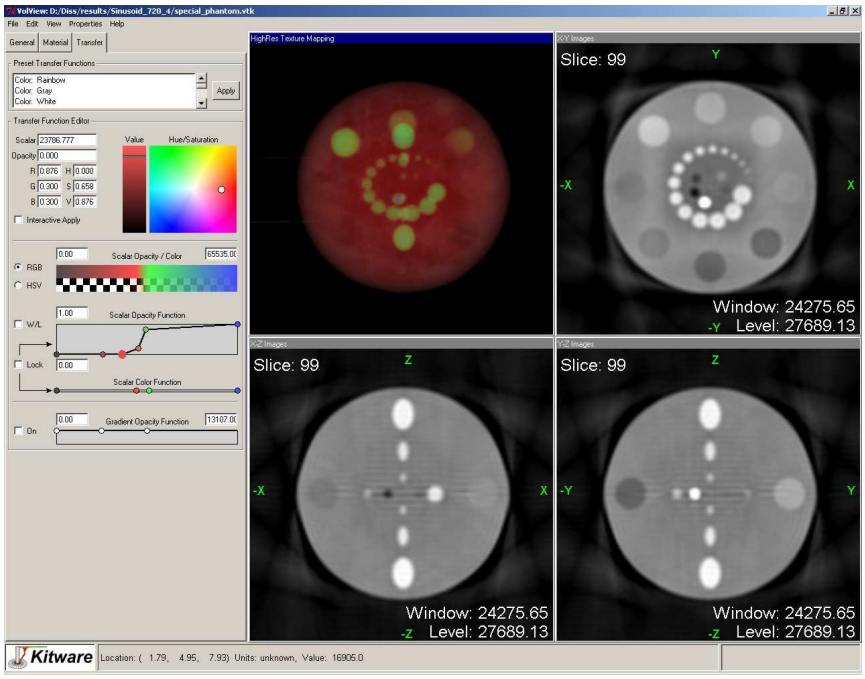

Abbildung 1.21: Rekonstruktion für die Sinusoid–Geometrie

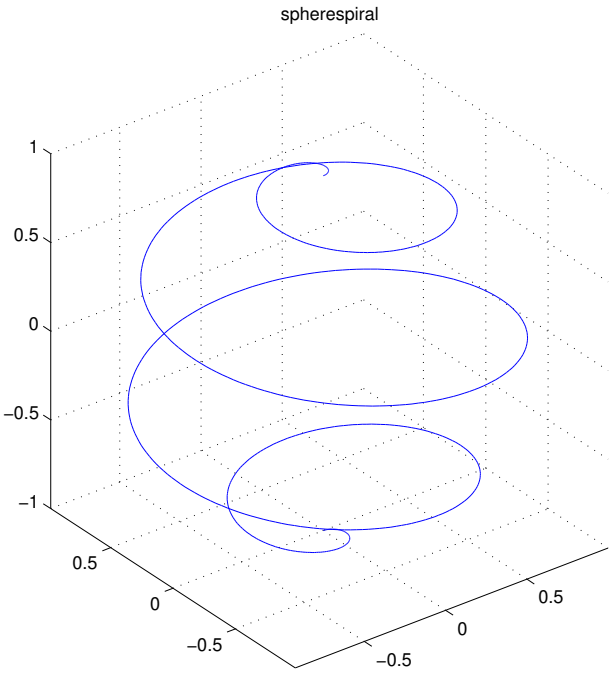

Abbildung 1.22: Sphärische-Spiral-Geometrie

Sphärische Spirale

Im Gegensatz zur Standardspiralgeometrie (siehe unten), deren Abtastpunkte sich auf dem Mantel eines Zylinders befinden, liegen hier die Punkte auf einer Kugeloberfläche verteilt. Ist die Anzahl der Umläufe mit p gekennzeichnet, so liegen die Punkte auf dem Weg

$$\Gamma(t) := K_2(\theta(t), \phi(t))$$

mit

$$\begin{aligned} \phi(t) &= 2pt\pi \\ \theta(t) &= t\pi \end{aligned}$$

Die Güte der Rekonstruktionen ist hier nicht so hoch wie bei den bisher betrachteten Geometrien. Grund dafür ist einerseits, dass gerade im interessanten Raum in der Nähe der $z = 0$-Schicht relativ wenige Projektionen verteilt sind und aufgrund der Diskretisierung

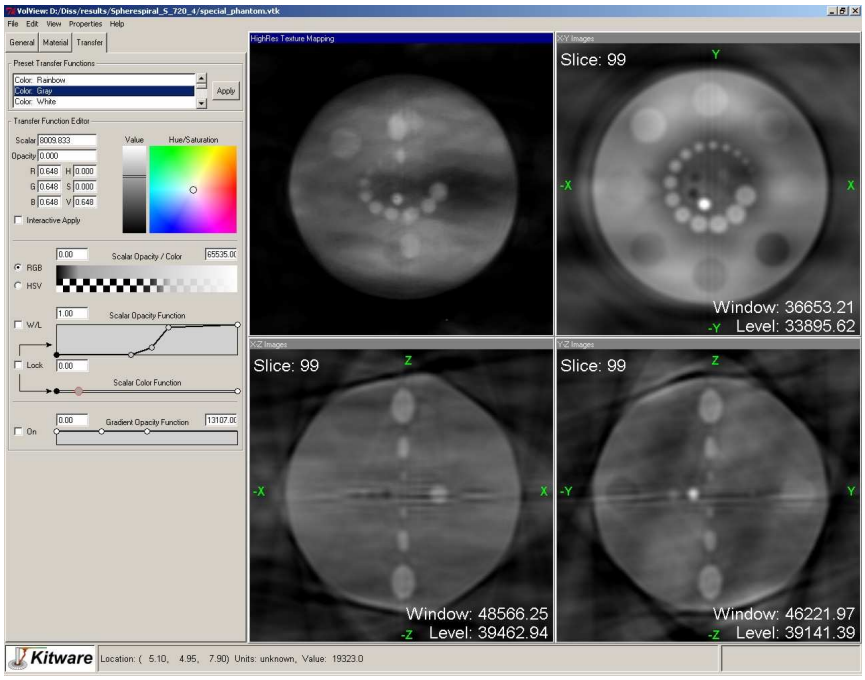

Abbildung 1.23: Rekonstruktion für die sphärische Spiral-Geometrie

viele Quellpunkte in der Nähe von z_max und z_min liegen. Dieses Ungleichgewicht drückt sich auch in der Form des Umgebungsellipsoids aus, das in x/z und y/z-Richtung deformiert ist. Die Hauptauswirkung der Artefakte drückt sich in einer geringeren Auflösung bei den Dichtesprüngen aus, die Grössenauflösung ist akzeptabel.

Eine Verbesserung der Ergebnisse ist durch eine Erhöhung der Umläufe und Projektionen, sowie durch eine nichtäquidistante Punkteverteilung auf dem Weg zu erreichen.

Spirale

Die Spirale ist eine für praktische Anwendungen interessante Geometrieanordnung, weil sie in Scannern leicht zu realisieren ist. Der Weg definiert sich mit p als Anzahl der Umläufe hier als

$$\Gamma(t) := K_3(\phi(t), 2t - 1)$$

mit

$$\phi(t) = 2pt\pi.$$

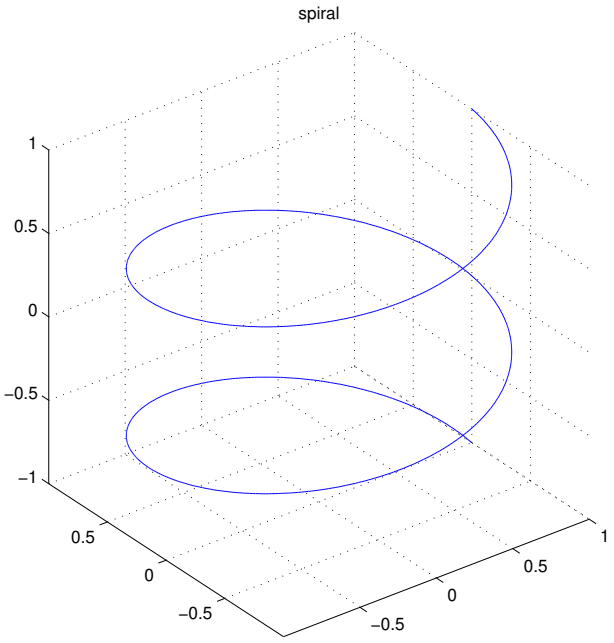

Abbildung 1.24: Spiral-Geometrie

Die Spiralgeometrie liefert im Bezug auf Auflösung von Dichtesprüngen und Ellipsoidgrösse von den betrachteten Tuy–Kirillov–Kurven die schlechtesten Ergebnisse (siehe Abb. 1.25). Ein Grund für die starken Diagonalartefakte ist die Tatsache, dass der Mittelstrahl durch das Objekt nicht mehr senkrecht auf dem Detektor auftrifft. Die senkrechte Ebene, in der die restlichen Artefakte konzentriert sind, kommt dadurch zustande, dass in dieser Ebene keine Quellpunkt zu finden sind. Sie zeigen eine ähnliche Struktur wie bei der später diskutierten Tomosynthese-Geometrie. Diese Form der Artefakte tritt nicht auf, wenn in

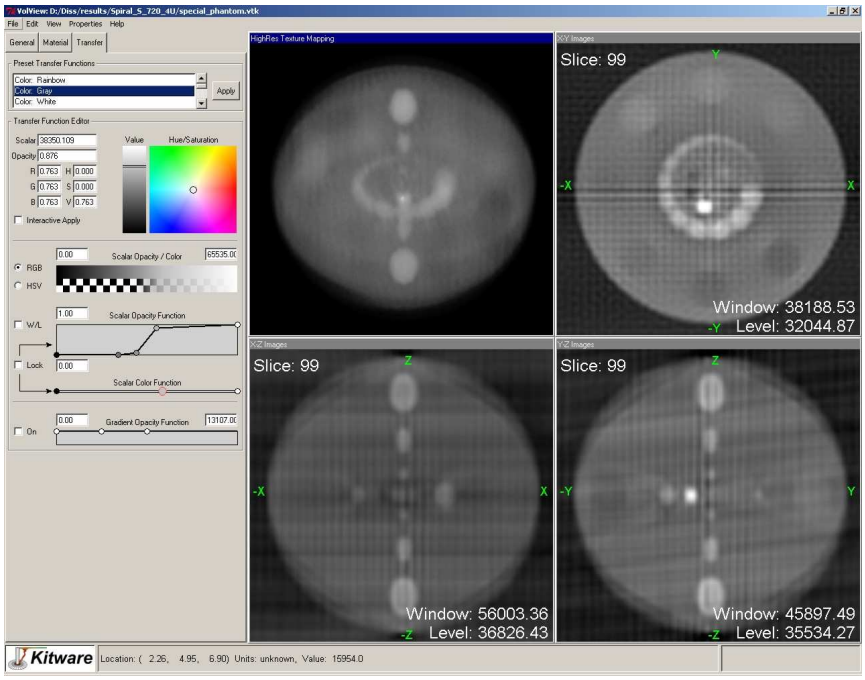

Abbildung 1.25: Rekonstruktion für die Spiral-Geometrie mit 4 Umläufen

der entsprechenden Ebene eine ausreichende Anzahl Quellpunkte zu finden ist, oder in einer Ebene senkrecht dazu. Im Fall der Spiralgeometrie müsste ein Umlauf in der x/z-Ebene oder in der x/y-Ebene erfolgen.

Die Güte der Rekonstruktionen ist durch eine Erhöhung der Umläufe und der Anzahl der Quellpunkte zu erzielen. Die Höhe des Abtastzylinders wurde hier so gewählt, dass das Objekt gerade noch vollständig auf dem Detektor liegt. Eine Vergrösserung des Zylinders, auch unter dem Risiko abgeschnittener Projektionen, kann u.U. das Ergebnis ebenfalls verbessern. Optimal wäre es, wenn der Mittelstrahl senkrecht auf dem Detektor auftrifft, was mit realen Scannern nicht leicht zu erreichen ist.

1.4.2.3 Andere Geometrien

Da die folgenden Geometrien die TUY–KIRILLOV–Bedingung verletzen, ist theoretisch keine exakte Inversion möglich.

Cone-Beam-Geometrie

Diese Geometrie ist die meist verwendete für alle Anwendungen in der 3D-Tomographie. Sie ist gerätetechnisch gut zu realisieren und die Güte der Rekonstruktionen ist sowohl für medizinische Anwendungen als auch für zerstörungsfreie Prüfverfahren ausreichend. Der Weg der Quelle ist einfach zu parametrisieren, es wird ein Vollkreis um das Objekt gefahren:

$$\Gamma(t) := d_R K_2(\pi/2, 2\pi t)$$

Bemerkung 1.4.1. Der Name dieser Geometrieanordnung ist irreführend, denn "Cone-Beam" ist, wie schon erwähnt, eine Strahlanordnung (die Alternative wäre "parallele Geometrie") und keine Bezeichnung für die Lage der Quellpositionen, doch er hat sich für diese Abtastgeometrie durchgesetzt und wird deshalb hier so übernommen.

Die Rekonstruktionen, die für diese Geometrie durchgeführt wurden, umfassen synthetische Daten (Abbildungen 1.26 und 1.27) und reale Daten (Abbildungen 1.28 bis 1.31). Ausserdem wurde bei dieser Geometrie der Einfluss der verschiedenen Basisfunktionen auf die Rekonstruktionsgüte getestet (Abbildungen 1.33 bis 1.36).

Die Rekonstruktionsgüte, gerade in der $z = 0$-Schicht, ist bei dieser Geometrieanordnung sehr gut. Grund ist die hohe Anzahl Quellpunkte in der x/y-Ebene, wodurch das Spektrum optimal abgedeckt wird. Probleme können ausserhalb dieser Ebene auftreten, wenn die Projektionen abgeschnitten sind, was im untersuchten Phantom allerdings nicht der Fall ist.

Auch die Versuche mit realen Daten zeigen sehr gute Ergebnisse, die sich von den mit gefilterter Rückprojektion mittels Approximativer Inverse visuell kaum unterscheiden (siehe Abbildung 1.32).

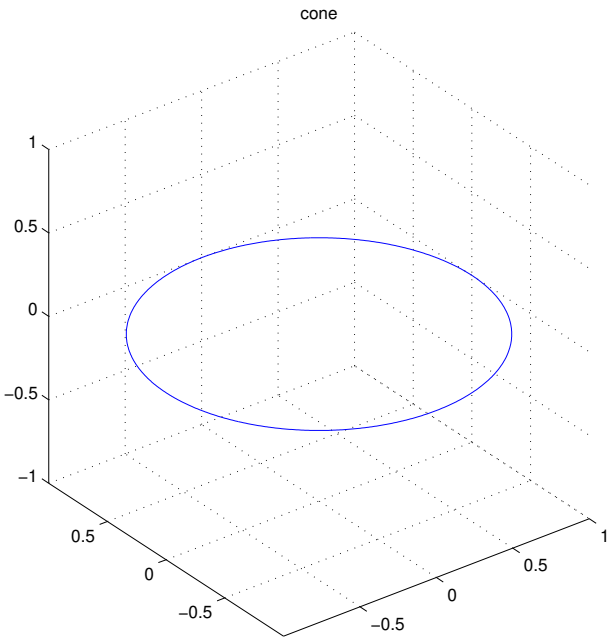

Abbildung 1.26: Cone-Beam-Geometrie

Was den Einfluss der verschiedenen Basisfunktionen angeht, so liefert die pyramidale Voxel-Basis die besten Ergebnisse, deshalb wurde sie für die vorherigen Tests verwendet. Die Gauss-Basis liefert geringfügig schlechtere Ergebnisse, erzeugt allerdings leichte vertikale Artefakte (innerhalb der Cone-Beam-Geometrie). Bei ihr wird auch etwas deutlicher, dass die zu rekonstruierende Funktion mit einer Basisfunktion gefaltet ist. Dieser Effekt ist bei der charakteristischen Funktion einer Kugel noch stärker. Bei ihr kommt hinzu, dass sie im Ortsraum steil abfällt, was im Frequenzraum zu Oszillationen führt. Daher ist der Filter abgeschnitten, was zusätzliche Artefakte verursacht. Diese Artefakte sind bei der charakteristischen Funktion des Würfels noch stärker. Die Basisfunktion dominiert vollkommen die Rekonstruktion, die dadurch unbrauchbar wird.

Der Einfluss der Strahlanordnung ist, wie Abbildung 1.36 zeigt, zu vernachlässigen. Die parallele Strahlanordnung, von der man eigentlich bessere Ergebnisse erwarten würde, liefert sogar geringfügig schlechtere Rekonstruktionen. Dies liegt daran, dass bei der Cone-Beam-Geometrie mit Cone-Beam-Strahlanordnung bei gleicher Detektorauflösung mehr Strahlen

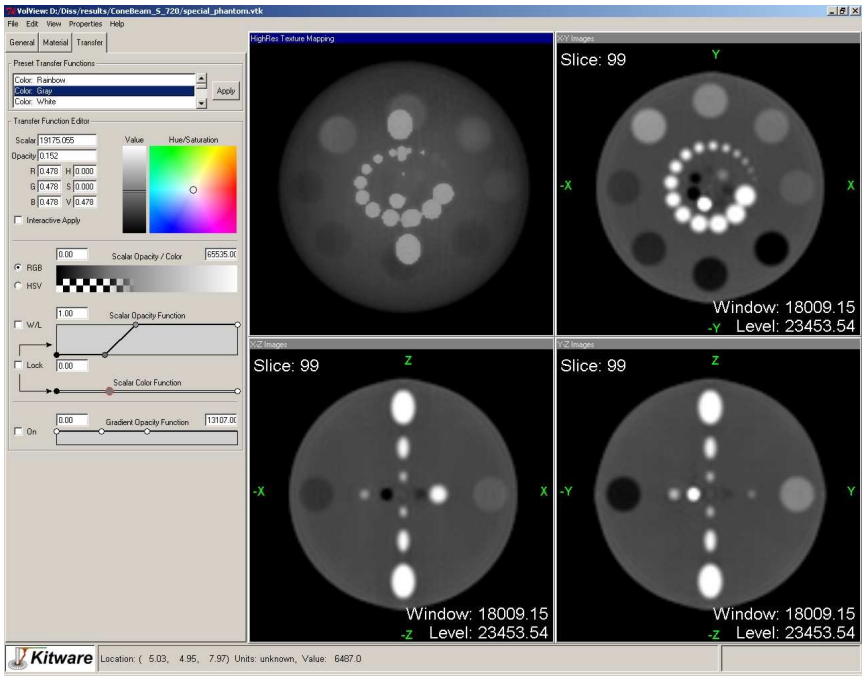

Abbildung 1.27: Rekonstruktion für die Cone–Beam–Geometrie (synth. Daten)

die interessante $z = 0$-Ebene und ihre Umgebung abdecken, während bei der parallelen Strahlanordnung ganz Ω gleichmässig abgetastet wird. Eine Auflösungserhöhung des Detektors bringt gleiche Resultate für beide Strahlanordnungen.

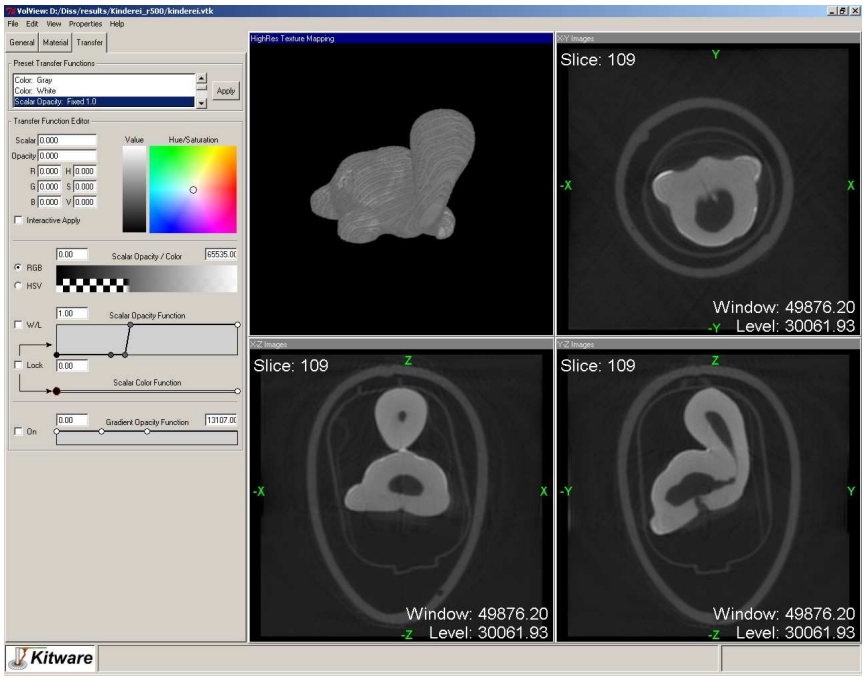

Abbildung 1.28: Rekonstruktion realer Daten: Basisgrösse 500

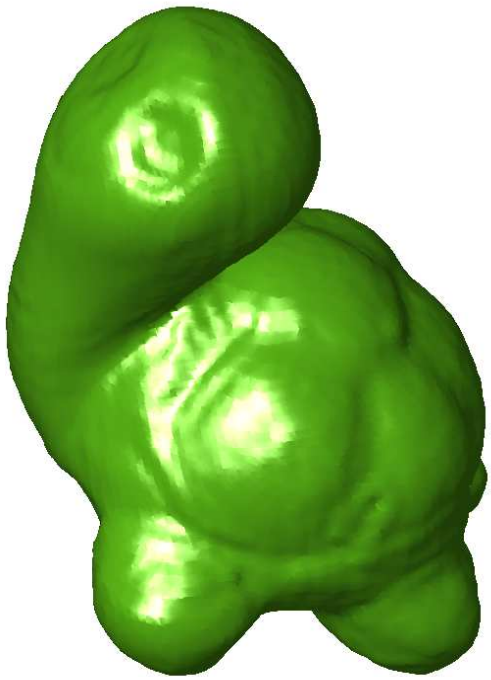

Abbildung 1.29: Rekonstruktion realer Daten: Basisgrösse 500, alt. Darstellung

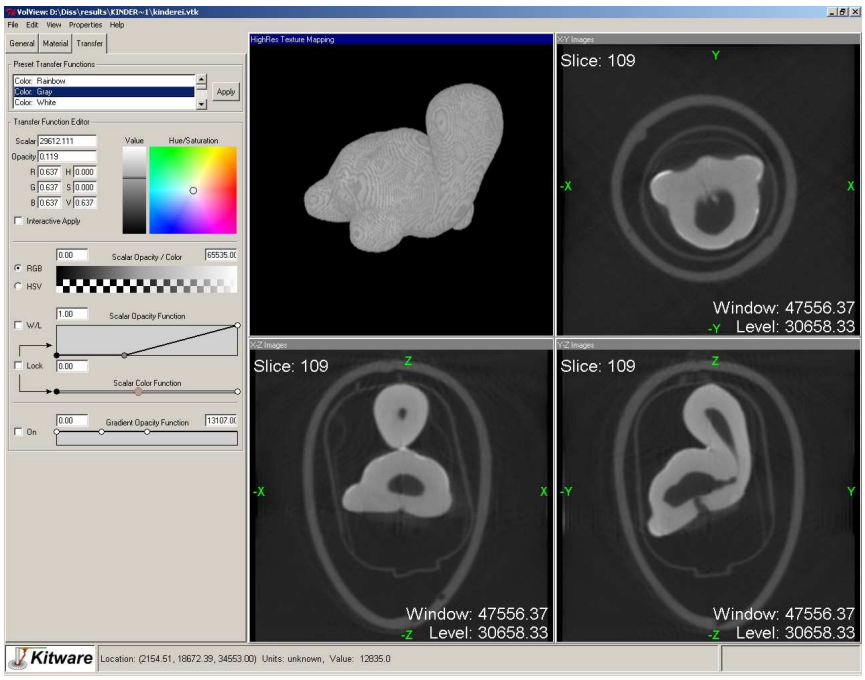

Abbildung 1.30: Rekonstruktion realer Daten: Basisgrösse 1000

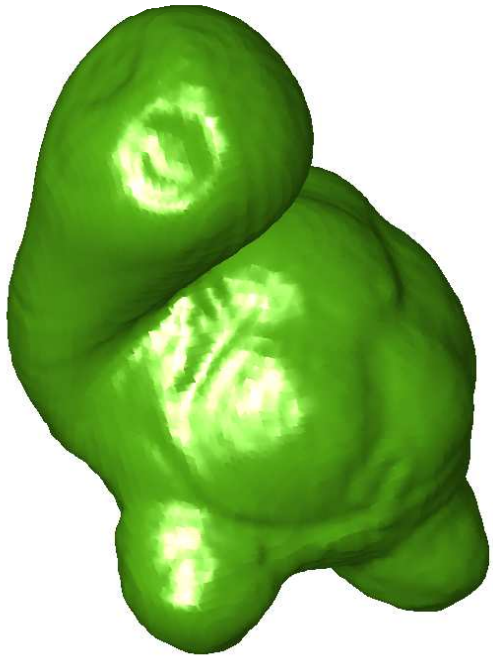

Abbildung 1.31: Rekonstruktion realer Daten: Basisgrösse 1000, alt. Darstellung

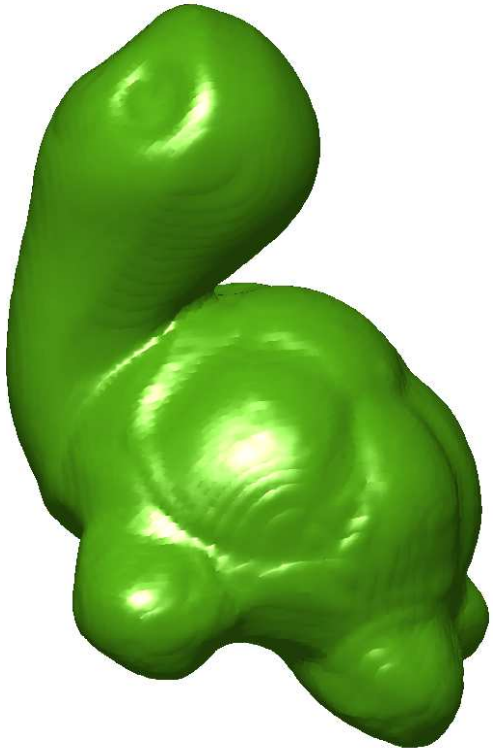

Abbildung 1.32: Rekonstruktion mit gefilterter Rückprojektion und Approximativer Inverse

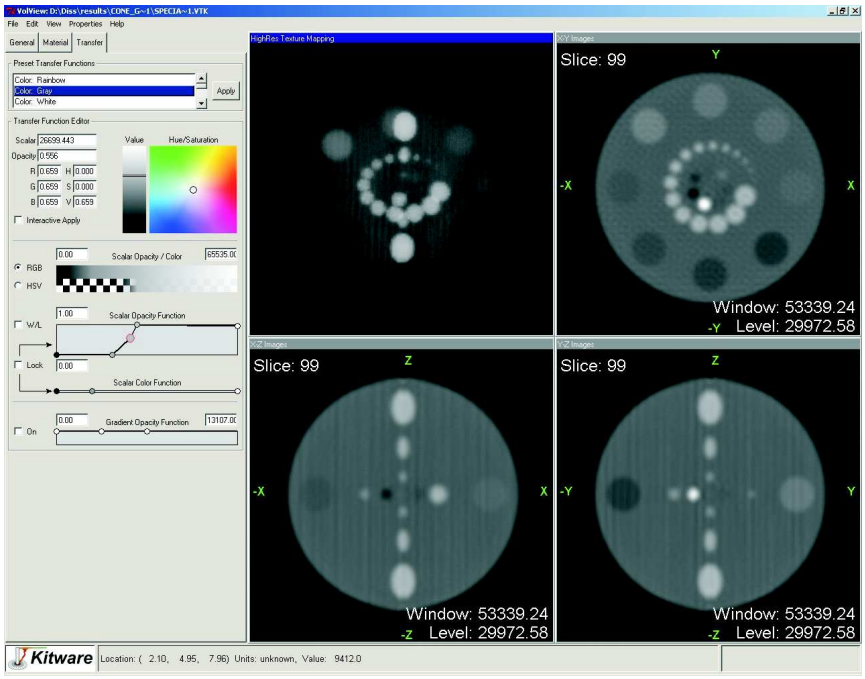

Abbildung 1.33: Rekonstruktion mit der Gauss-Basisfunktion

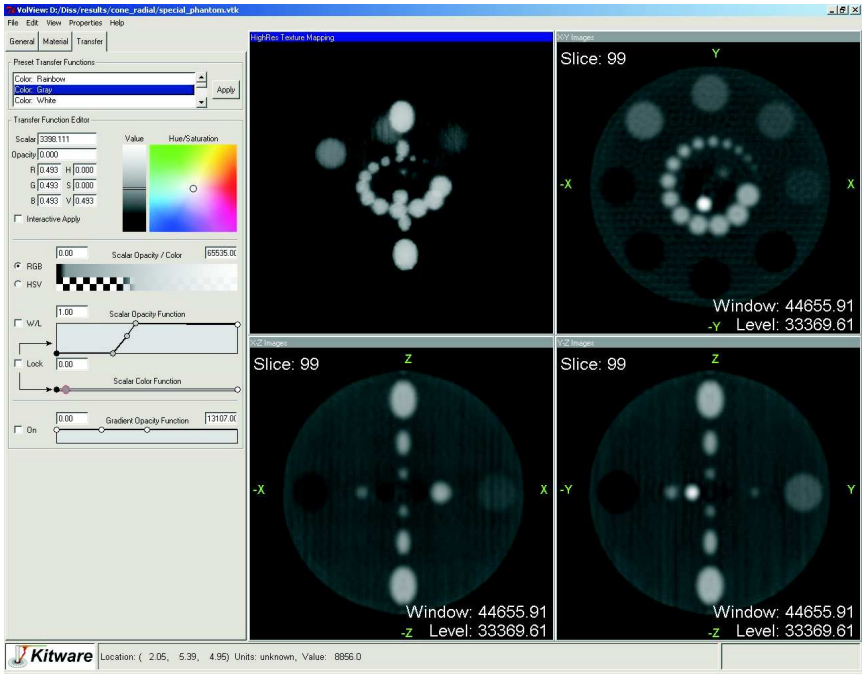

Abbildung 1.34: Rekonstruktion mit der radialen χ-Basisfunktion

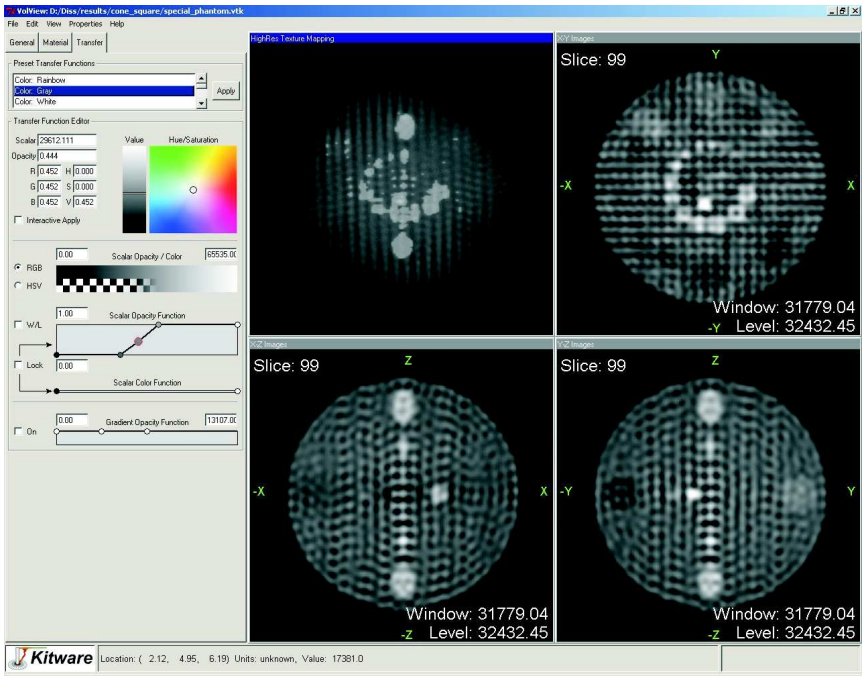

Abbildung 1.35: Rekonstruktion mit der Würfel-χ-Basisfunktion

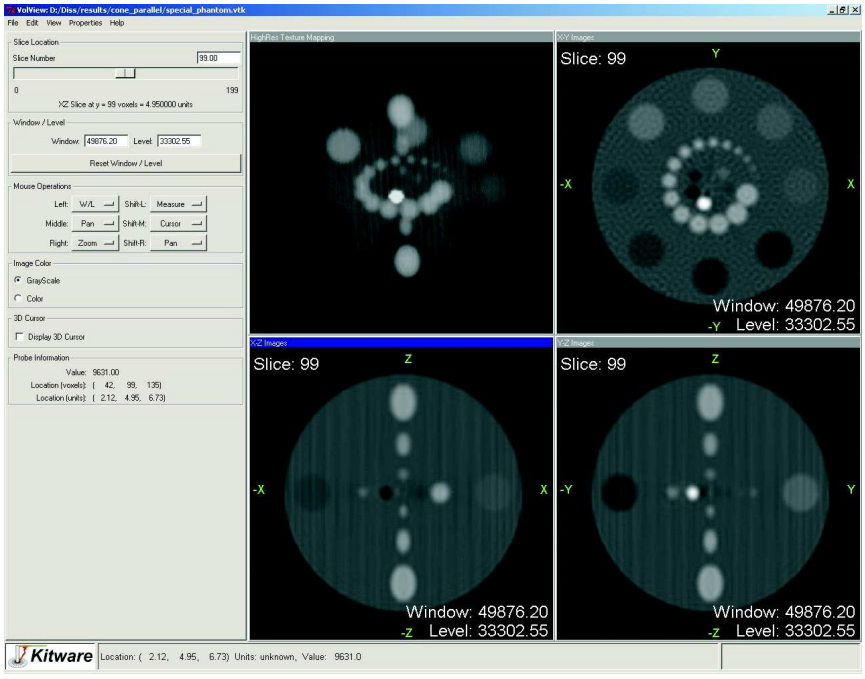

Abbildung 1.36: Rekonstruktion mit paralleler Geometrie und pyramidaler Basisfunktion

Tomosynthese

Diese besonders im zahnmedizinischen Bereich gebräuchliche Geometrieanordung ist von der Inversion her die schlechteste, denn hier wird noch nicht einmal ein Vollkreis, sondern nur ein Kleinkreis um bzw. über das Objekt gefahren. Gerade die Tiefenauflösung leidet dadurch erheblich und der einigermassen gut rekonstruierbare Bereich beschränkt sich bei herkömmlichen Verfahren deshalb grösstenteils auf die sogenannte Fokalebene, den Brennpunkt der Aufnahmen, sowie etwas darüber und darunter. Je weiter man sich von der Fokalebene entfernt, umso stärker werden die Artefakte. Ist man nur am oben genannten Bereich interessiert, so ist die Tomosynthese ein durchaus praktikables Verfahren.

Die Abtastkurve ist ähnlich der bei der Cone-Beam-Geometrie, jedoch statt $\theta = \pi/2$ wird hier $\theta = \alpha/2$ gewählt, wobei α der sogenannte Tomosynthesewinkel, also der Öffnungswinkel gegen die e_3-Achse ist. Im Grenzfall $\alpha = \pi$ entspricht diese Geometrie dann der Cone-Beam-Geometrie. Es ergibt sich folgender Weg

$$\Gamma(t) := d_R K_2(\alpha/2, 2\pi t)$$

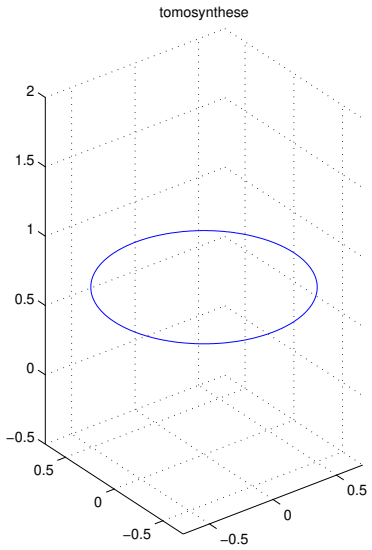

Abbildung 1.37: Tomosynthese-Geometrie

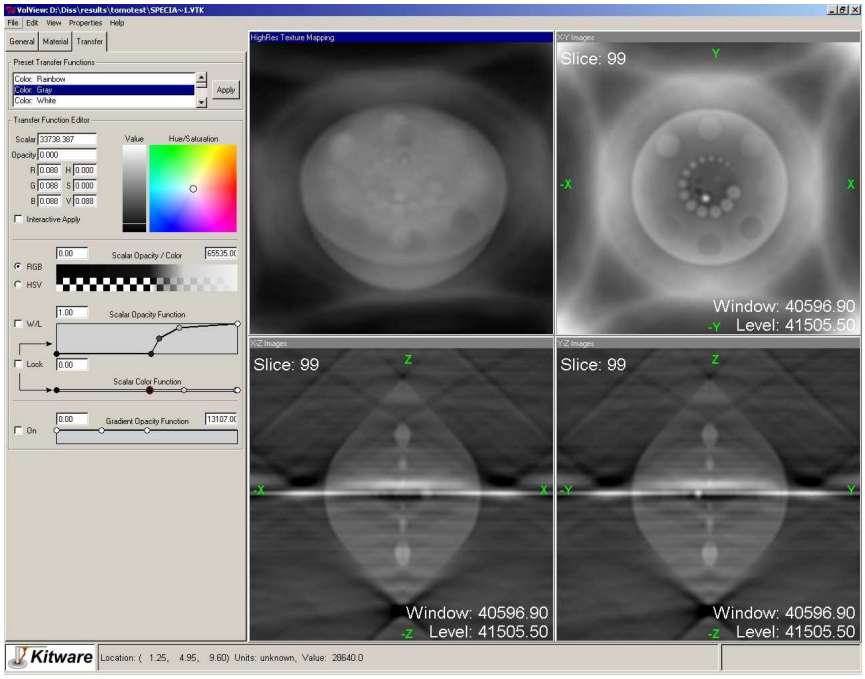

Abbildung 1.38: Rekonstruktion für die Tomosynthese-Geometrie

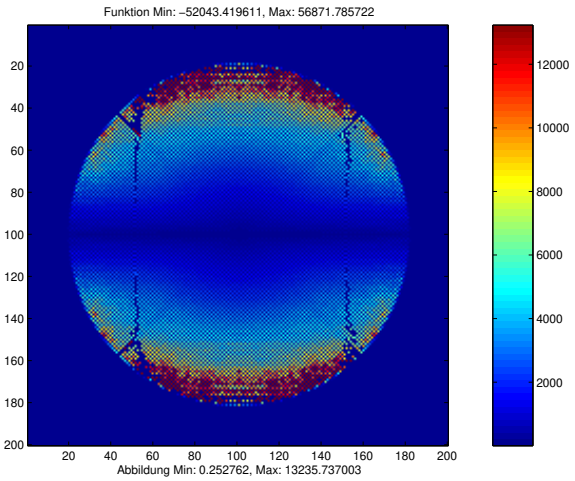

Abbildung 1.39: Filter für Conebeam, xy-Schnitt

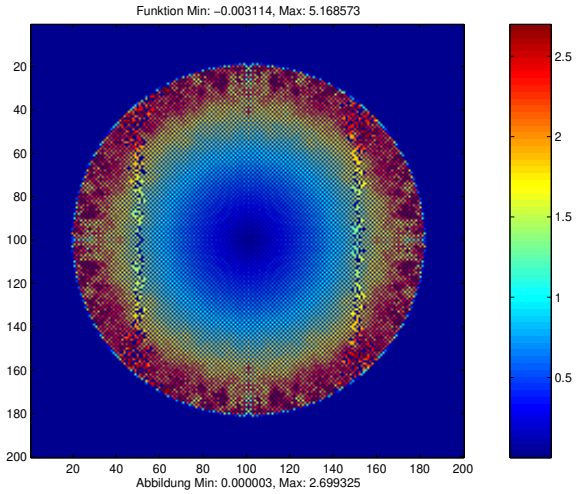

Abbildung 1.40: Filter für Tomosynthese, xy-Schnitt

Was die Güte der Rekonstruktionen betrifft, sind interessanterweise sind die Artefakte um die Fokalebene konzentriert. Trotzdem kann man die Ellipsoide klar erkennen. Die Bereiche, die normal nicht rekonstruiert werden können, also im Abstand von der Fokalebene, sind relativ deutlich dargestellt, wenn auch mit den typischen Limited-Angle-Deformationen. Eine Betrachtung des Spektrums des verwendeten Filters H_Γ im Vergleich mit dem der Cone-Beam-Geometrie (CBG) zeigt wesentliche Unterschiede (siehe Abb. 1.39 bis 1.44): während im xy-Schnitt noch ein Profil zu erkennen ist, auch wenn es vom CBG-Profil stark abweicht, besteht ein Grossteil der yz- und xz-Schnitte aus Lücken. Die Informationen in diesen Bereichen sind bei der Filterberechnung nicht vorhanden und werden deshalb bei der Rekonstruktion nicht gewichtet. Dies führt zu einem sehr dünnen Frequenzspektrum der resultierenden Funktion und die Artefakte verteilen sich aufgrund des globalen Charakters der Fourier-Transformation über einen grösseren Bereich des Ortsraums.
Eine Verbesserung der Rekonstruktionen ist durch Vergrösserung des Tomowinkels möglich, was allerdings dem Ziel tomosynthetischer Rekonstruktionen widerspricht.

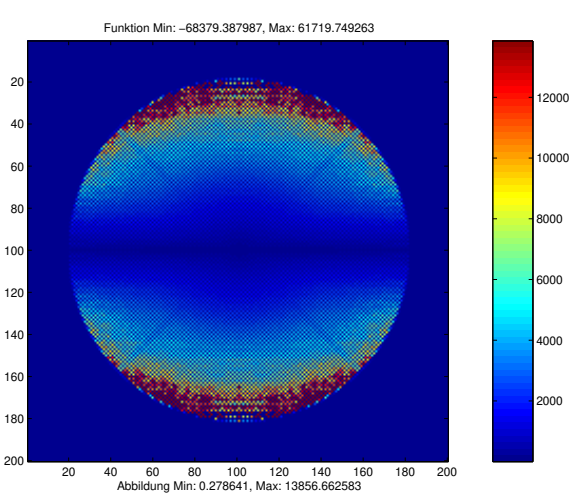

Abbildung 1.41: Filter für Conebeam, xz-Schnitt

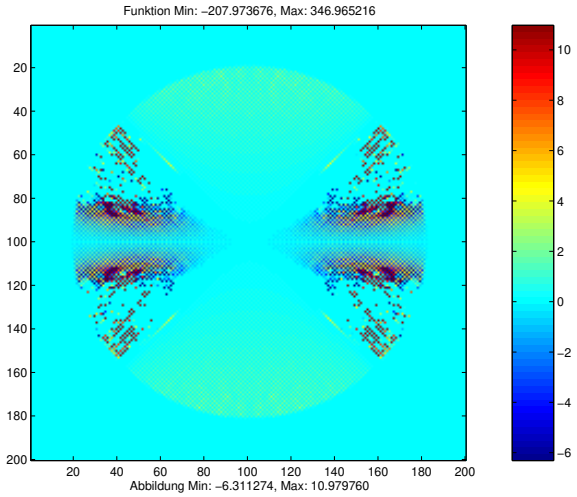

Abbildung 1.42: Filter für Tomosynthese, xz-Schnitt

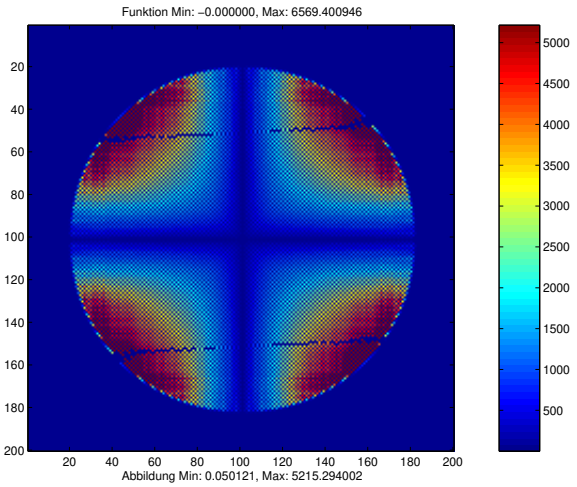

Abbildung 1.43: Filter für Conebeam, yz-Schnitt

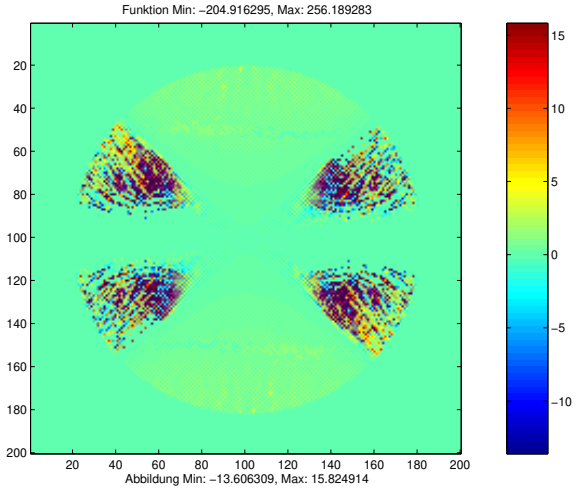

Abbildung 1.44: Filter für Tomosynthese, yz-Schnitt

Kapitel 2

Implementierung:
Das TORNE Framework

2.1 Einführung

In der Computertomographie spielt neben der mathematischen Erforschung neuer oder verbesserter Rekonstruktionsalgorithmen die Implementierung eine entscheidende Rolle. Während im 2D-Fall aufgrund der Datenlage (eindimensionale Daten, zweidimensionales Rekonstruktionsgebiet, einfache Geometrien) noch eine Behandlung mittels prozeduraler Programmiersprachen wie FORTRAN und Rapid-Development-Umgebungen wie Matlab möglich ist, so verlangt die 3D-CT doch erheblich mehr Resourcen in Hinsicht auf Prozessorzeit und Speicheranforderungen. Der dadurch ebenfalls wachsenden Komplexität Algorithmen und der Verwaltung dieser Resourcen muss mittels verbesserten Implementierungs- und Programmiertechniken entgegengesteuert werden.

In typischen Fällen kann man bei der Rekonstruktion von folgenden Eckdaten ausgehen:

Parameter	2D-Fall	3D-Fall
Datengrösse (Richtungen*Detektorelemente)	$170 * 512 \cong 680KB$	$120 * 1024^2 \cong 960MB$
Rekonstruktionsgebiet	$512^2 \cong 2MB$	$512^3 \cong 1024MB$
Anzahl Geometrieparameter	< 10	> 30
Quellcodezeilen (C/C++)	> 300	> 4000
Laufzeit	$O(n^3)$	$O(n^4)$

Durch die ersten beiden Punkte ist die Entscheidung für eine prozessornahe Programmiersprache schon implizit vorgegeben. Die Anzahl Parameter und die komplexen Datenstrukturen verlangen ein entsprechendes Design der Programme. Unter Berücksichtigung beider Argumente wurde der Sprache C++, entwickelt von Bjarne Stroustrup und inzwischen Standardisiert, der Vorzug gegeben, da diese durch ihre Einbettung des prozessornahen C schnelle und durch ihre objektorientierte Grundstruktur komplexe Programmierung unterstützt.
Beschäftigt man sich mit angewandter Forschung, so ergeben sich weitere Anforderungen an die Implementierung:

1. schnelle Entwicklungszyklen: Erfordernis für eine schnelle Umsetzung neuer Algorithmen ist die geringstmöglichste Modifikation schon geschriebener Softwareteile. Gerade im Bereich angewandter Forschung ändern sich die Problemstellungen und Anforderungen schnell. Daher sollte die Reaktionszeit auf Änderungen so kurz wie möglich sein. Man kann diesen Punkt als Hauptanforderung betrachten und die nachfolgenden als Spezifizierung der Methoden, mit denen er erreicht wird.

2. Wiederverwendbarkeit: Bestehende Programmteile sollten so weit wie möglich genutzt werden können, ohne dass eine tiefgreifende Neuimplementation notwendig ist.

3. adäquate Behandlung der Problemkomplexität: Die grosse Anzahl an (z.B. Geometrie-)Parametern, die Teil der 3D-CT sind, erfordert eine Implementation, die eine übersichtliche und effiziente Parametrisierung gestattet.

4. Erweiterbarkeit von Algorithmen: Bestehende Algorithmen sollten leicht erweiterbar und modifizierbar sein.

5. einfache Anbindung realer Daten und passender Visualisierungsmethoden: So sinnvoll synthetische Daten zur Erprobung neuer Algorithmen auch sind, letztendliches Ziel der Programmierung ist die Behandlung realer Daten. Der Import dieser Daten muss einfach zu implementieren und auch schnell auf sich ändernde Datenformate anpassbar sein. Zur Analyse der Güte neuer Verfahren muss darüberhinaus eine Anbindung an Visualisierungssoftware einfach geschehen können.

Die objektorientierte Programmiertechnik (OOP) erfüllt die Anforderungen 1. - 4. auf natürliche Art und Weise, Punkt 5. ist dann innerhalb der Programmstruktur einfach umzusetzen. Da die OOP die Einzelteile einer Implementierung entkoppelt, ist eine Erweiterung problemlos möglich, ohne dass diese sich auf das Gesamtsystem auswirkt. Die Vorteile, die diese Art der Softwareentwicklung aufweist, machen es allerdings notwendig, das Problem aus einer ihr eigenen Sichtweise zu analysieren und ein Design zu erstellen.
Im Rahmen des Designs fallen nun immer wieder auftretende Grundstrukturen auf, deren Gesamtheit eine abstrakte Beschreibung von Algorithmen ermöglicht. Sie bilden die Grundlage für das Klassenframework TORNE – ein Akronym für *TOmographic Reconstruction NEtwork* – und seien im weiteren *mathematische Entwurfsmuster* genannt. Interessant ist hier, dass diese Beschreibung auf eine Vielzahl inverser Probleme anwendbar ist.
Die Struktur dieses Kapitels ist wie folgt: zu Beginn in Abschnitt 2.2 sei eine kurze, allgemeine Einführung in die OOP gegeben, da diese nicht notwendigerweise zum Grundrepertoire numerischer Programmierung gehört. Der nächste Abschnitt 2.3 stellt das abstrakte Design des vorliegenden Softwarepaketes in den Mittelpunkt und stellt somit eine Anwendung des ersten Abschnitts dar. In Teil 2.4 findet man die konkrete Beschreibung verschiedener Rekonstruktionsverfahren der 3D-Computertomographie und Tomosynthese, insbesondere des GUF-Verfahrens aus 1.3, innerhalb des TORNE-Frameworks. Eine Beschreibung aller mathematischen Entwurfsmuster und der auf ihnen basierenden Objekte, den sogenannten Muster- und Objektkatalog kann dem Anhang 3.1 entnommen werden.

2.2 OOP und Muster

2.2.1 Programmierparadigmen

Programmiertechniken lassen sich anhand ihrer Paradigmen, d.h. den ihnen zugrundeliegenden Denkmodellen, klassifizieren. Programmiersprachen unterstützen ein bestimmtes Paradigma, wenn sie 'Elemente zur Verfügung stellen, die das Programmieren in diesem Stil nahelegen' [41]. Die folgende Darstellung orientiert sich am Programmentwurf (im Gegensatz zu formalen Definitionen).

Eine kurze Klassifikation verschiedener Paradigmen ist wie folgt:

Prozedurales Programmieren

Paradigma: Man unterteile ein Programm in gewünschte Prozeduren und verwende dann den besten Algorithmus.

Dieses Paradigma wurde in der Sprache FORTRAN zum ersten mal umgesetzt. Es ist ablauforientiert, da der Algorithmus die zentrale Rolle spielt. Jede Prozedur zeichnet sich durch wohldefinierte Eingabe- und Rückgabegrössen aus, was eine nahe Anlehnung an die mathematische Formulierung einer Funktion darstellt. Daher ist diese Programmiertechnik für Programme in mathematischem Kontext noch weit verbreitet, solange die Betonung auf den Algorithmen und nicht auf den Daten liegt. Verlagert sich der Schwerpunkt auf die Organisation von Daten, dann ist vereinfacht sich die Implementierung durch

Modulares Programmieren

Paradigma: Man entscheide, welche Module gebraucht werden und zerlege das Programm so, dass die Daten in Module eingekapselt sind.

Innerhalb der Module gilt weiterhin die prozedurale Programmiertechnik. Die Module selbst sind allerdings in sich abgeschlossen und eine Modifikation von Daten muss über Funktionen in den Modulen geschehen. Variablen, Konstanten, Funktionen und Typen sind lokal für das Modul. Im Idealfall stellt können die Module einzeln entworfen und kompiliert werden. Somit ist dieser Stil zur Bearbeitung grösserer Projekte geeignet, die von einem Team entwickelt werden.

Jedes Modul stellt im Prinzip einen neuen Datentypus dar, also eine Erweiterung eingebauter Typen wie Integer, Float und Character, und es sollten Funktionen zum Anlegen, Löschen und Bearbeiten zur Verfügung gestellt werden. Dies ist ein grosser struktureller Vorteil, aber es ist nicht unbedingt ausreichend, da diese Typen für den Compiler eine Objekt*beschreibung* darstellen und nicht etwa ein neues Objekt. Aspekte wie die Erzeugung verschiedener Instanzen eines Objektes, Konstruktion und Destruktion können zwar umgesetzt werden, doch nicht mit Sprach- und Compilerunterstützung. Daher ist hier ein weitergehendes Konzept notwendig, nämlich das der

Datenabstraktion

Paradigma: Man entscheide, welche Typen gebraucht werden und stelle eine komplette Menge von Operationen für jeden Typ zur Verfügung.

Hier ist jeder neue, benutzerdefinierte Datentyp eine Erweiterung eingebauter Datentypen und geniesst die volle Compilerunterstützung, d.h. es wird z.B. automatisch erkannt, wenn ein Objekt erzeugt, aber nicht gelöscht wurde. Beispiel für einen neuen Datentyp in mathematischem Kontext sind komplexe Zahlen, die als Komposition von Fliesskommavariablen mit zugehörigen Zugriffsstrukturen implementiert werden können und somit "die Sprache erweitern".

Was diesem Paradigma fehlt, ist Flexibilität: sobald ein benutzerdefinierter Datentyp modifiziert wird, müssen alle Zugriffsstrukturen angepasst oder ein neuer Datentyp eingeführt werden. Einen Ausweg bildet

Objektorientiertes Programmieren

Paradigma: Man entscheide, welche Klassen gebraucht werden und stelle für jede Klasse eine komplette Menge an Operationen zur Verfügung. Gemeinsamkeiten werden über Vererbung explizit gemacht.

Vererbung bildet die Grundlage der Abstraktion. Sobald zwischen Objekten Gemeinsamkeiten bestehen, werden diese abstrahiert und in einer Basisklasse spezifiziert. Von dieser werden die speziellen Klassen abgeleitet, d.h. die Funktionalität der Basisklasse wird erweitert oder so modifiziert, dass sie den Bedürfnissen entspricht.

Beispiel 2.2.1. In dem später verwendeten Klassenframework TORNE ist die Klasse CDetector abgeleitet von CContainer2D (siehe 3.1), da ein Detektor (als mathematische Modellierung) eine zweidimensionale Matrix (die wiederum einen Container darstellt) mit zusätzlichen Eigenschaften ist.

Der grosse Vorteil einer solchen Vorgehensweise besteht darin, dass in abgeleiteten Objekten nur die Unterschiede zur Basisklasse spezifiziert werden müssen und die Gemeinsamkeiten schon als Attribute übernommen werden.
Die nichttriviale Aufgabe, vor der man bei diesem Programmierparadigma steht, ist das Design, d.h. die Herausarbeitung von Gemeinsamkeiten und Unterschieden in der Gesamtheit der benötigten Objekte.

Die eigentliche Implementierung wurde, wie schon erwähnt, in der Sprache C++ durchgeführt, da diese sowohl prozessornahe Programmierung (über die eingebettete Sprache C) ermöglicht, als auch die notwendigen Erweiterungen zum objektorientierten Programmieren enthält. Eine komplette Sprachreferenz findet sich in [41].
In den nächsten Abschnitten werden die Begriffe der folgenden Definition 2.2.1 benötigt. Für eine alternative Kurzfassung siehe z.B. [37].

Definition 2.2.1. Begriffe der objektorientierten Programmierung:

- Objekt

  Ein Objekt ist ein eigenständiges Individuum mit charakteristischen Eigenschaften, einem inneren Zustand, Attributen und Methoden.

- Klasse

  Ein Klasse ist die Vereinigung und formale Beschreibung aller Objekte mit gleichen Eigenschaften. Klassen können als Implementation abstrakter Datentypen aufgefasst werden. Klassen werden auch als Objektfabriken bezeichnet.

- Instanz

  Eine Instanz ist die konkrete Realisierung eines Objektes einer Klasse. Instanzen sind genau ein Objekt mit einem Gültigkeitsbereich. Sie werden durch einen Konstruktor erzeugt und durch einen Destruktor zerstört.

- Attribut

  Attribute sind die Variablen, die den inneren Zustand eines Objekts beschreiben. Sie korrespondieren in ihrer Bedeutung mit den Sorten des abstrakten Datentyps.

- Methode

  Methoden sind die über einer Klasse erklärten und ihr zugeordneten Operationen. Ein Methodenaufruf ist Teil einer Nachricht, die an ein Objekt versandt wird. Methoden sind auch als Operationen bekannt.

- Vererbung

  Vererbung ist die Weitergabe von Eigenschaften (Attributen und Methoden) einer Basisklasse an eine abgeleitete Klasse. Sie existiert in verschiedenen Ausprägungen:

  - Spezialisierung: Objektmenge der Unterklasse ist Teilmenge der Objektmenge der Oberklasse.
  - Typhierarchie: jedes Objekt des Untertyps verhält sich wie ein Objekt des Obertyps
  - Klassenhierarchie: Vererbung der Implementation

  Weitere Details zu Vererbung sind in Abschnitt 2.3.2 dargestellt.

- Schnittstelle

  Eine Schnittstelle ist die Gesamtheit aller durch die Operationen eines Objektes definierten Methoden, genauer deren Parameter, Methodenname und Rückgabewert.

- Polymorphie

  Polymorphie ist die Möglichkeit, Objekte mit identischen Schnittstellen bei Auswertungen von Methoden (ggf. bei der Laufzeit) zu substituieren.

- virtuelle Funktionen

  Virtuelle Funktionen erlauben es, in der Basisklasse Funktionen zu deklarieren, die in den abgeleiteten Klassen redefiniert werden. Es ist Aufgabe des Compilers, für den Aufruf einer virtuellen Funktion die entsprechende Version auszuwählen. Bei der Programmierung werden virtuelle Funktionen durch Funktionsdeklaration innerhalb des Header-Files (ggf. unter Anwendung des Schlüsselwortes *virtual*) ohne assoziierten Funktionsrumpf gekennzeichnet.

- abstrakte Basisklasse

  Abstrakte Basisklassen verbieten durch den Einsatz virtueller Funktionen die Erzeugung konkreter Instanzen. Sie stellen ein Konzept oder eine Schnittstellendefinition dar. Ein Beispiel dafür sind die Basisklassen, die von den Mustern von TORNE erzeugt werden. Man kann abstrakte Basisklassen als die Konkretisierung dieser Muster ansehen.

Nachdem die Begriffe und Konzepte des OOP eingeführt sind, soll hier noch kurz auf die Bedeutung für das numerische Rechnen eingegangen werden. Wie schon zu Beginn erläutert, wächst in der angewandten Mathematik nicht nur die Datenmenge, sondern es werden aufgrund verbesserter Möglichkeiten durch die Rechnerentwicklung auch komplexere Probleme (bzw. bekannte Probleme mit komplexeren Methoden) behandelt. Dazu ist Teamarbeit notwendig, die von der OOP sehr profitiert, denn die Implementation wird dabei in Objekte mit wohldefinierten Schnittstellen zerlegt, von denen jedes unabhängig von allen anderen umgesetzt werden kann.
Doch auch bei kleineren Projekten liegen die Vorteile auf der Hand. Man überlege sich als Beispiel einen Löser für ein lineares Gleichungssystem $Ax = b$. Wird dieser mittels OOP konzipiert, dann ist es z.B. ohne weiteres möglich, hinter dem Objekt der Matrix

A eine vollbesetzte oder eine dünnbesetzte Matrix zu "verstecken", ohne dass dies dem Löser bekannt ist (angewandte Polymorphie), denn dieser operiert auf der Schnittstelle der Matrixklasse. Man kann also die Problembehandlung sukzessive verfeinern, ohne dabei das Gesamtkonzept zu modifizieren oder schon bestehende Implementationsteile neu behandeln zu müssen.
Dabei ist, wie oben schon angedeutet, das Design wichtig, das schnittstellenbasiert sein muss. Die Schnittstelle der Objekte muss die Anforderung der Vollständigkeit, der möglichen Polymorphie und der hinreichenden Abstraktion erfüllen. Schnittstellen sind somit das Kernthema des Designs, auf dessen Grundprinzipien im nächsten Abschnitt eingegangen wird (konkrete Designfragen des Projekts TORNE sind in Abschnitt 2.3 zu finden).

2.2.2 OO-Modellierung und Design

Das Design eines Programms, d.h. die Modellierung in Objekte, ist von grundlegender Bedeutung für das OOP. Während bei prozeduralen Programmiersprachen durch die Ablauforientiertheit in gewisser Hinsicht der Rahmen der Programmierung vorgegeben ist (ein mehr oder minder sequentielles Abarbeiten von Funktionsaufrufen), so erfordert das OOP mehr Vorarbeit. Ziele eines guten Softwaredesigns sind die folgenden:

- Robustheit

  Die Software sollte Ausnahmebedingungen (Auftreten eines Fehlers) abfangen und konsistent arbeiten. Eine robuste Implementierung sichert somit Programmierern ein effizientes Arbeiten.

- Verständlichkeit

  Das Design sollte eine logische Struktur aufweisen, die auch für Projektfremde schnell verständlich ist. Dies wird u.a. von sinnvollen Objektbezeichnungen unterstützt.

- Erweiterbarkeit

  Das System sollte es ermöglichen, bei geänderten Rahmenbedingungen leicht erweiterbar und modifizierbar zu sein.

- Modularität

  Modularität bedeutet eine geringstmögliche Verknüpfung der Komponenten des Systems, was die anderen hier genannten Ziele unterstützt.

- Wiederverwendbarkeit

  Wiederverwendbarkeit bedeutet die Einbettung von Objekten eines Projektes in andere Projekte, die ein ähnliches Softwareproblem lösen.

- Wartungsfreundlichkeit

  Die Wartung, also Fehlerbeseitigung, Ergänzung von Algorithmen, etc., ist ein wichtiger Punkt. Fehlerbeseitigung in einem Teil der Software sollte keine Fehler in einem anderen hervorrufen.

Das Zerlegen der Software in Objekte ist eine teilweise diffizile Aufgabe. Zu dessen Bewältigung hat sich die Object Modeling Technique von Rumbaugh et al [36] bewährt. Dabei werden drei Teilmodelle verwendet, die das Design spezifizieren. Beispielanwendungen sind in Abschnitt 2.4 zu finden. Auf eine Einführung der Diagrammnotation wird hier aus Platzgründen verzichtet. Die Modelle sind wie folgt:

- Objektmodell

  Es dient dazu, alle Objekte mit ihren Eigenschaften und Beziehungen zu anderen Objekten innerhalb des Systems zu identifizieren. Dieses statische Modell bildet oft die Hauptzerlegung des Designs.

- Dynamisches Modell

  Das dynamische Modell bildet die Laufzeitdynamik ab. Es enthält mögliche Abläufe und Interaktionen.

- Funktionales Modell

  Das funktionale Modell zeigt den Datenfluss durch das System und die stattfindenden Modifikationen. Somit enthält es die funktionalen Abhängigkeiten der Einzelprozesse

und Objekte. Die grundlegende Unterteilung besteht in Sources, Data Sinks und Processes (Originalterminologie), die in abgewandelter Form als Provider, Processes und Outlets in TORNE definiert werden. Das funktionale Modell spielt u.a. bei der Parallelisierung eine grössere Rolle.

2.2.3 Entwurfsmuster

Beim Design von Software, insbesondere in ähnlichem Problemkontext, sind oft Gemeinsamkeiten abstrakter Funktionalität der gefundenen Objekte auffällig. Daher ist es sinnvoll, noch eine Abstraktionsebene höher zu gehen und nach allgemein auftretenden Mustern beim Design zu suchen. Dies führt in grösseren Projekten zu den sog. Entwurfsmustern, die erstmals von Gamma et al [10] systematisiert und katalogisiert wurden. Die dort angegebene Definition lautet:

Definition 2.2.2. Ein *Entwurfsmuster* benennt, motiviert und erläutert systematisch einen allgemeinen Entwurf, der ein in objekt-orientierten Systemen immer wiederkehrendes Entwurfsproblem löst.

Während in [10] als Musterelemente Mustername, Problemabschnitt, Lösungsabschnitt und Konsequenzenabschnitt genannt werden, sind diese im mathematischen Kontext grösstenteils durch die Problemstellung vorgegeben. Auf diese wird in der Darstellung der *mathematischen Muster* daher verzichtet.
Die Muster werden nach ihrer Aufgabe grob unterteilt in

- Erzeugungsmuster: ihre Aufgabe ist die Objekterzeugung
- Strukturmuster: sie enthalten die Zusammensetzung von Klassen und Objekten
- Verhaltensmuster: sie definieren, wie Objekte zusammenarbeiten und wie die Zuständigkeiten aufgeteilt werden.

Der Entwurfsmusterkatalog, der in [10] angegeben wird, ist sinnvoll bei der Implementierung grösserer Softwaresysteme. Die Anwendung in mathematischem Kontext ist zwar

möglich, aber nicht unbedingt notwendig. Die allgemeinen Strukturen, die beim Systementwurf von Software zur Lösung inverser Probleme, von denen die 3D-Tomographie eines darstellt, auszumachen sind, die *mathematischen Muster*, werden nach anderen Kriterien klassifiziert, nämlich nach den Teilobjekten, die innerhalb des zu lösenden Problems auftreten. Trotzdem erfüllen sie Definition 2.2.2 , wenn man die Klasse der zu lösenden Probleme auf den mathematisch-numerischen Kontext beschränkt. Ein Beschreibung der mathematischen Muster findet sich in Abschnitt 2.3.1; der komplette Musterkatalog ist Bestandteil von 3.1.
Beim Design fallen mathematische Muster als abstrakte Basisklasse auf, die sich oft dadurch auszeichnen , dass die angewandten Methoden innerhalb abgeleiteter Klassen genauer spezifiziert, d.h. überschrieben, werden müssen.

Beispiel 2.2.2. Ein *Container* muss zwingenderweise einen Zugriff auf die in ihm enthaltenen Elemente besitzen. Diese Zugriffsmethode kann sich je nach Art der Attribute in Rückgabewert und Zugriffsparameter erheblich unterscheiden. Somit stellt der Container an sich ein mathematisches Muster dar, aber keine instanzierbare Klasse.

Eine Überlappung zwischen Entwurfsmustern laut Gamma und mathematischen Mustern existiert durchaus, eine eins-zu-eins Abbildung gestaltet sich jedoch schwierig, da erstere auf eine wesentlich grössere Klasse von Problemen, nämlich allen, die durch ein Softwaresystem gelöst werden können, zugeschnitten sind. Die Grobklassifizierung in Erzeugungsmuster, Strukturmuster und Verhaltensmuster kann jedoch auch bei mathematischen Mustern angegeben werden (siehe Abschnitt 2.3.1).

2.3 Design

2.3.1 Die Muster in TORNE

Bei der vorliegenden Arbeit war ein Teilziel die geschickte Implementierung von Rekonstruktionsverfahren in der 3D-CT. Das Design vollzog sich dabei in mehreren Schritten, die jetzt im einzelnen beschrieben werden.
Um *passende Objekte* zu finden, muss als erstes das Problem beschrieben werden. Betrachtet man existierende Rekonstruktionsverfahren in der 3D-CT, so fallen diese in folgende Grobklassen:

- *Gefilterte Rückprojektion*

  Dieses Verfahren hat in Operatornotation die Form $f = \mathcal{D}^* F \mathcal{D} f$, wobei $\mathcal{D} f$ die Daten repräsentiert, F einen Filteroperator darstellt und $\mathcal{D}^*$ den adjungierten Operator zu $\mathcal{D}$, welcher auch als Rückprojektion bezeichnet wird und die Abbildung zweidimensionaler Daten in einen dreidimensionalen Raum darstellt. Die Filterung selbst wird aus Zeitgründen hauptsächlich im Fourier-Raum durchgeführt, so dass man davon ausgehen kann, dass praktisch $F = \mathcal{F}^{-1} \widehat{F} \mathcal{F}$ umgesetzt wird.

  Diese Rekonstruktionsmethode ist aufgrund ihrer Genauigkeit die, in der Praxis medizinischer und technischer Anwendungen, favorisierte.

- *Filterung nach Rückprojektion* Gemäss Abschnitt 1.3.1 beschrieben, hat diese Rekonstruktionsmethode,welche auch als ρ-filtered-layergram bezeichnet wird, die Form $f = \bar{F} \mathcal{D}^* \mathcal{D} f$, der Filterungsprozess $\bar{F}$ wird nach der Rückprojektion durchgeführt.

- *GUF-Verfahren*

  Das GUF-Verfahren laut 1.3 stellt in gewissem Sinn ein Hybridverfahren aus den ersten beiden dar und kann beschrieben werden als $\widetilde{f} = H_e \mathcal{D}^* \widetilde{F}_e \mathcal{D} f$. Zu den Details siehe das entsprechende Kapitel.

- *Direkte Fouriermethoden*

  Diese Rekonstruktionstechnik nützt das Fourier-Slice-Theorem aus, das wie schon vorher angedeutet, für die Kegelstrahlgeometrie nicht existiert. Prinzipiell ergibt sich für ein solches Verfahren die Form $f = \mathcal{F}^{-1}PF\mathcal{F}Rf$, wobei hier F die Ableitung im Fourier-Bereich und die Projektion P als Abbildung von 2D nach 3D ein Eintragen von Frequenzinformationen des 2D-Spektrums an passender Stelle im 3D-Spektrum mit nachfolgender Interpolation von Polar- zu kartesischem Gitter darstellt.

- *Iterative Verfahren*

  Während im 2D-Fall iterative Verfahren eine gewisse Bedeutung besitzen, so sind sie im 3D-Fall eher von theoretischem Interesse, da die Bearbeitung der grossen Datenmengen dabei zu kostspielig ist. Sie seien daher (auch wenn das folgende Design ihre Implementation unterstützt) bei der Betrachtung ausgeklammert.

Analysiert man obige die Rekonstruktionsverfahren, so stellt man fest, dass sie in Teilabschnitte zerlegbar sind, die immer der Form $f = Ag$ genügen, d.h. man operiert mittels der Abbildung A auf Daten g, um das Resultat f zu erhalten. Diese, an sich triviale Erkenntnis, ist nichtsdestotrotz Grundlage der Implementierung, da sie schon die Grobforderung des Klassenframeworks enthält: *es soll die Bearbeitung beliebiger strukturierbarer Daten mittels Abbildungsprozessen gestatten und unterstützen.* Betrachtet man nun die Form, in der sich diese Prozesse aneinanderreihen, so ergibt sich die Form eines Netzwerks mit meist linearem Verlauf, ähnlich dem, welches komplexerer Visualisierungssoftware zugrundeliegt (siehe [39]). 3D-Visualisierung bildet auf ihre Art das duale Problem der 3D-Tomographie, nämlich die Abbildung von dreidimensionalen Strukturen auf ein zweidimensionales Medium, wodurch sich gewisse Parallelen übertragen lassen.
Was wir also als Grundmuster erkennen, ist die Aufteilung in CContainer und CProcess (Abbildung 2.1).

Bemerkung 2.3.1. Die hier verwendeten Bezeichnungen entstammen der später verwendeten Klassennotation (das C am Anfang steht gemäss üblicher Konvention für 'Class').

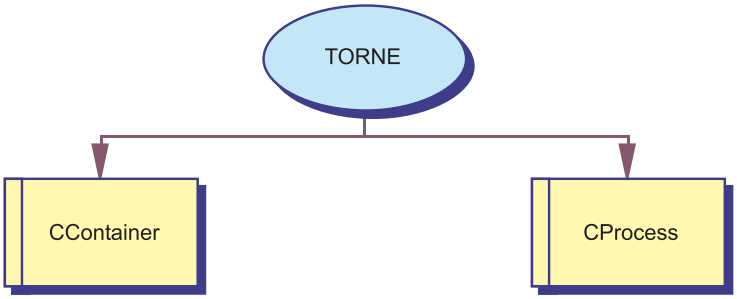

Abbildung 2.1: Grundstruktur von TORNE

Das Containermuster kapselt dabei die Daten, der Prozess die Abbildungen. Die später verwendeten Klassen bieten für den Container alle notwendigen Zugriffsstrukturen, während die Prozesse alle Abbildungseigenschaften tragen, die nicht an den Container gebunden sind.

Beispiel 2.3.1. Betrachten wir nun im folgenden die gefilterte Rückprojektion als Beispiel möglicher Rekonstruktionsverfahren. Um zu implementationsfähigen Objekten zu kommen, ist die Funktionaldarstellung aufgrund ihrer abstrakten Formulierung (die zur Mustergewinnung durchaus von Vorteil ist) weniger aussagekräftig. Stattdessen benötigt man eine diskrete Version, zu der man wie folgt gelangt:
Die Funktionaldarstellung

$$f_\gamma = \mathcal{D}^* F_\gamma \mathcal{D} f$$

wird über die Integraldarstellung

$$f_\gamma(x) = \int_\Gamma \int_{S^2} \mathcal{D}f(a,\theta) k_\gamma(x;a,\theta) d\theta da$$

in die diskrete Darstellung

$$f_\gamma(x) = c \sum_{l=0}^{P-1} \sum_i \sum_j k_{m(x;a,l)-i),n(x;a,l)-j,\gamma} Df_{i,j,l}$$

überführt. Aus letzterer lassen sich dann folgende Einzelschritte extrahieren:

1.

$$Df_{l,\gamma}(x) = \sum_i \sum_j k_{m(x;a,l)-i),n(x;a,l)-j,\gamma} Df_{i,j,l}$$

Dieser Schritt stellt die Anwendung eines Faltungsfilters auf eine zweidimensionale Matrix dar.

2.
$$\forall y \in \Omega : f_{l,\gamma}(y) = Df_{y,l,\gamma}$$

Die Rückprojektion bildet 2D-Daten auf einen 3D-Kubus ab.

3.
$$f_\gamma = \sum_l f_{l,\gamma}$$

Integration über die Richtungen erfolgt als Summation der Einzelrückprojektionen.

Damit stehen als Containerobjekte bisher zur Diskussion:

- die Daten: zweidimensionale Matrizen
- das Rekonstruktionsgebiet: dreidimensionaler Voxel-Kubus
- die Grössen- und Geometrieparameter

Die Prozesse lassen sich folgendermassen formulieren:

- Filterung: Abbildung von 2D nach 2D
- Rückprojektion: Abbildung von 2D nach 3D
- Richtungssummation: Abbildung von 3D nach 3D

Beenden wir hier das Beispiel.

Analysen der obigen Form sind ein erster Schritt zur Klassifizierung der benötigten Objekte. Verfolgt man den Ansatz weiter, so gelangt man zu einem Satz von Mustern, der wie folgt aussieht: [Diagramme 2.2 und 2.3].

Definition 2.3.1 (Musterdefinition). Die Hauptmuster unterteilen sich in Container und Process. Bei ersteren unterscheiden wir:

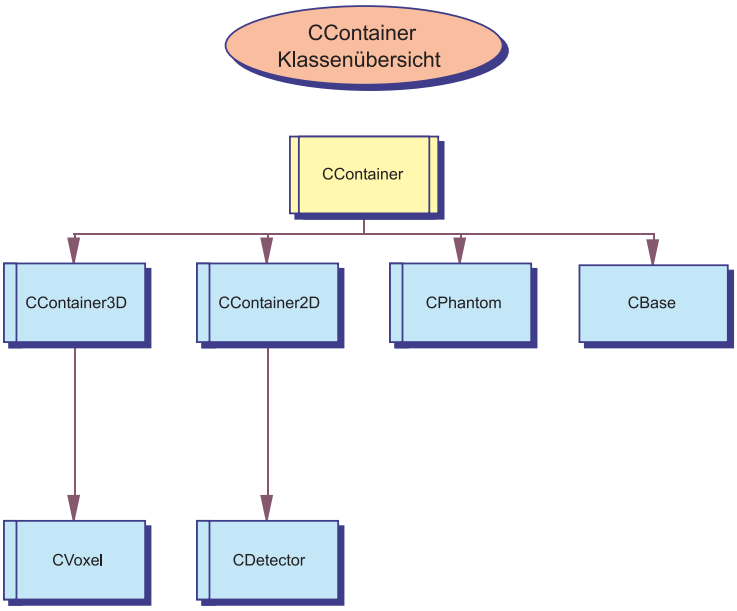

Abbildung 2.2: CContainer

- *Matrixäquivalente Container*: Sie bilden die grundlegende Struktur, denn sowohl die Daten als auch Filter und Rekonstruktionsgebiet sind im Diskreten Matrizen. Bsp: CContainer2D, CVoxel.

- *Freistrukturierte Container*: Zu dieser Form von Containern zählt zum Beispiel die Beschreibung der verwendeten Phantome der Rekonstruktion. Auch wenn man die entsprechenden Daten durchaus in Matrizen repräsentieren könnte (was mit allen Daten möglich ist), so wäre diese Darstellung entweder redundant oder unübersichtlich. Es widerspricht somit einer natürlichen Darstellung. Bsp: CPhantom.

- *Analytische Container*: Beispiel solcher Container sind die im GAF-Verfahren verwendeten Basiselemente, die innerhalb des Verfahrens als Matrizen betrachtet werden. Da diese Matrizen allerdings von vielen Parametern abhängen, sind die entsprechenden Daten durch eine analytische Repräsentation wesentlich leichter zu handhaben, da sie dann 'on the fly' erzeugt werden können und in der Zwischenzeit wenig Resourcen benötigen. Es handelt sich also quasi um Diskretisierungsobjekte, die auf

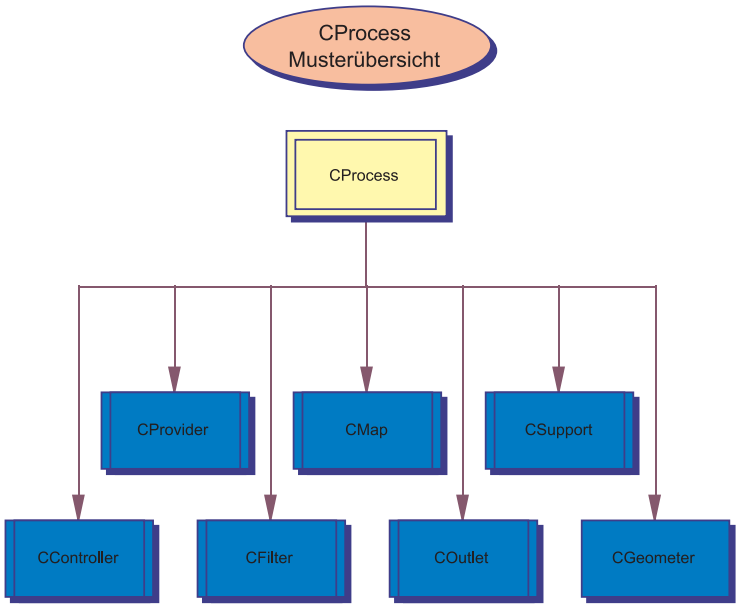

Abbildung 2.3: CProcess

eine analytische Funktion bezogen sind. Bsp: CBase.

Die Prozesse unterteilen sich wie folgt:

- Kontrollmuster: Diese Muster repräsentieren das Rekonstruktionsverfahren und die verwendete Geometrie (wobei die Abhängigkeiten so weit wie möglich getrennt sind). Ihre Repräsentanten sind:
  - *Controller*: Ein Controller implementiert das Rekonstruktionsverfahren bezüglich *einer* Projektion, d.h. in ihm sind alle Prozesse zusammengefasst, die zur Behandlung einer Projektion notwendig sind.
  - *Geometer*: Der Geometer beschreibt die Geometrie und iteriert über einen Controller, d.h. er setzt alle Parameter, die für eine bestimmte Projektion spezifisch sind und ruft den Controller mit diesen Parametern auf.

  Bemerkung 2.3.2. Controller und Geometer sind das "Herz" eines Rekonstruktionsverfahrens. Im Beispiel der gefilterten Rückprojektion sind bis auf die Berechnung

der Geometrie alle Prozesse innerhalb des Controllers zu finden. Für die ρ-filtered-layergram-Methode ist der Controller-Algorithmus sehr kurz und der Geometer enthält die wichtigen Filterprozesse.

In der Notation der Entwurfsmuster [10] kommen die Kontrollmuster den Strukturmustern nahe.

- Ein- / Ausgabe: Diese Muster sind, wie der Name schon sagt, die Grundlage für die Bereitstellung von Daten und ihrer letztendlichen Weitergabe an den Benutzer. Hier findet man:
  - *Provider*: stellen Daten zur Verfügung. Dabei kann es sich z.B. um synthetische Daten oder reale Daten, die auf einem Sekundärspeicher bereit liegen, handeln.
  - *Outlet*: exportieren Daten. Der Export ist in herkömmlichen Sinn das Schreiben in eine Datei. Man kann ohne weiteres jedoch auch *direkt* Visualisierungsoftware wie z.B. das vtk-toolkit (siehe [39]) anbinden, so dass sich der Rekonstruktionsalgorithmus in die Visualisierungspipeline einbindet.

  Diese Muster entsprechen im weitesten Sinn den Erzeugungsmustern in [10].

- Abbildungen: Als letzte und in der Anzahl der abgeleiteten Objekte zahlenmässig grösste Klasse existieren die Abbildungsmuster. Die Klassifikation lautet:
  - *Filter*: die Filter stellen Abbildungen unter Erhaltung von Dimension und Typus der angebundenen Container dar, d.h. ein 2D-Filter wird als Eingabecontainer eine 2D-Matrix erwarten und auch eine solche weitergeben. Einzige Ausnahme für eine etwas weiter gefasste Interpretation dieser Definition sind die FFT-Klassen, die eine Matrix u.U. von $\mathbb{R}^n$ nach $\mathbb{C}^n$ abbilden. Hier betrachte man $\mathbb{R}$ in $\mathbb{C}$ eingebettet.
  - *Mapper*: ein Mapper stellt eine Abbildung mit Veränderung von Dimension und/oder Typus des Containers dar. Ein Beispiel für einen Mapper ist die Rückprojektion.

Muster dieses Typus kann man grob mit den Verhaltensmustern in [10] gleichsetzen.

Bemerkung 2.3.3. Obige Muster bilden lediglich einen Grundstock für die Bearbeitung tomographischer Rekonstruktionen und verschiedener inverser Probleme. Wird ein neues Verfahren implementiert, dann werden wenn möglich bestehende Klassen verwendet. Sind keine entsprechenden vorhanden, wird eine neue von einem bestehenden Muster abgeleitet. Ist kein passendes Muster vorhanden, muss das Framework an sich erweitert werden. Durch die abstrakte Formulierung sollte zumindest letzteres in jedem Fall möglich sein.

2.3.2 Weitergehende Designfragen

Der vorliegende Abschnitt behandelt Designfragen, die sowohl aus programmiertechnischen Gesichtspunkten kommen, als auch aus der praktischen Implementation.
Als ersten Punkt des weiteren Designs muss man sich die Frage der *Objektgranularität* stellen. Im Prinzip kann man jede verwendete mathematische Funktion als eigenen Prozess definieren. Wollte man dies jedoch mit Funktionen wie z.B. der *sgn* (Signums-)Funktion, der Berechnung von Skalarprodukten oder der Betragsfunktion umsetzen, so würde dies sowohl zu einer Klassenexplosion innerhalb des Frameworks (d.h. den zur Verfügung stehenden Objekten), als auch zu einem sehr unübersichtlichen Bearbeitungsnetz führen. Deshalb wurde das Framework um ein weiteres Prozessmuster erweitert, dem *Supporter-Muster (CSupport)*. Man kann dieses Muster als eine Library an Funktionen ansehen, die für einen eigenen Prozess zu klein sind. Hier wird die Stringenz des Frameworks durchbrochen, doch aus praktischen Gründen ist eine solche Massnahme erforderlich. Jede Klasse kann selbstverständlich eine eigene Supporter-Klasse besitzen und es ist möglich, Supporter-Klassen für verschiedene Anwendungsbereiche zu implementieren.
Desweiteren ist der Punkt der *Objektschnittstellen* zu klären. Jeder Prozess hat von seiner ursprünglichen Definition her eine Input- und eine Outputschnittstelle, die als Typ einen Container akzeptieren und den Datenfluss beschreiben. Ausserdem existiert eine Parent- und eine Childschnittstelle, die an einen anderen Prozess angekoppelt werden dürfen (oder müssen) und die Ablaufdynamik regeln.

Bemerkung 2.3.4. Um Endlos-Rekursionen zu vermeiden, wird bei der Ablaufkontrolle ein Reference-Counter verwendet.

In den vom Muster abgeleiteten Objekten können die Schnittstellen dann so definiert werden, dass zur Compile-Zeit ein korrekter Aufbau des Verarbeitungsnetzwerks garantiert wird (strenge Typenkontrolle), so dass z.B. an einen 2D-Filter auch nur 2D-Objekte angebunden werden.
Was die *Implementierung von Objekten* betrifft, ist zu sagen, dass die Entwurfsmuster als abstrakte Basisklassen definiert werden, d.h. als Klassen, die mindestens eine virtuelle Funktion besitzen. Sie bieten lediglich die Grundfunktionalität, die aus der Definition des Musters abgeleitet werden kann.
Das Muster CProcess definiert beispielsweise die im letzten Absatz genannte Basisschnittstelle und stellt die Methode 'Link' zur Verfügung, die einen Prozess in Bezug auf Datenstrom und Ablauf mit einem anderen verbindet.
In jeder Verfeinerung des Musters, der Vererbung, werden dann zusätzliche generische Methoden implementiert, sobald man davon ausgehen kann, dass diese in abgeleiteten Klassen genutzt werden. Beispiel dafür ist die 'Apply'-Methode des Filters, die einen Filter konkret anwendet. Die Ablaufkontrolle wird in einem späteren Abschnitt behandelt werden.

Bemerkung 2.3.5. Die hier genannte Art der Vererbung ist als Klassenvererbung, d.h. als Vererbung von Code und Repräsentation, bekannt. Ihr gegenüber steht die reine Schnittstellenvererbung, bei der keine implementierte Methode innerhalb der abstrakten Basisklasse zu finden ist. Eine solche würde voraussetzen, dass die Basisklasse alle Methoden als 'pure virtual' deklariert.

Die exakte Behandlung der letzten Punkte erfolgt aus der Notwendigkeit heraus, *schnittstellenbasiert zu programmieren.* Wenn man den Vorteil, den Polymorphie bietet, wirklich anwenden will, und dies ist Voraussetzung für Wiederverwertbarkeit von Softwareelementen, dann muss es ein Ziel sein, auf eine Schnittstelle hin zu programmieren. Die Rettung von bestehenden Implementationsteilen kann schon durch die Technik der Vererbung gesichert werden; wesentlicher jedoch ist die Erweiterbarkeit des Frameworks dadurch, dass

Schnittstellen intelligent definiert werden. Sie führen dazu, dass Teile des Systems komplett unabhängig von anderen Teilen modifiziert und ergänzt werden können.
Mit der Wiederverwertbarkeit eng verknüpft, ist die Unterscheidung von *Klassenvererbung* (oder White-Box-Vererbung) und *Objektkomposition* (Black-Box-Vererbung). Bei ersterer wird die Funktionalität bestehender Klassen durch direkte Vererbung erhalten. Dies führt dazu, dass Attribute und Methoden der Basisklasse offenliegen und zugänglich sind. Bei zweiterer wird die Funktionalität durch reine Benutzung der definierten Schnittstelle wiederverwendet, interne Strukturen dieser Klasse bleiben verborgen. Deswegen ist im Allgemeinen letzterer Zugang zu favorisieren. Im TORNE-Framework kommen beide Formen der Wiederverwendung vor. Die Klassenvererbung wurde hauptsächlich dann angewandt, wenn ein direkter Zugriff auf die interne Repräsentation der Daten aus Effizienz- oder Berechnungsgründen notwendig war.

Beispiel 2.3.2. Die von CContainer2D ableitete Klasse CDetector wird mit der zusätzlichen Methode 'GetC' ausgestattet, die einen Zugriff auf bilinear interpolierte Werte innerhalb des Detektorgebietes ermöglicht, während 'Get' über Integerparameter logische Elemente adressiert. Objektkomposition hätte in diesem Fall dazu geführt, dass pro Anfrage zwei 'Get'-Aufrufe ausgewertet werden müssen. Durch Klassenvererbung kann auf die Datenelemente von CContainer2D zugegriffen werden und somit die Effizienz der Auswertung wesentlich gesteigert werden.

Bei einer Erweiterung von TORNE sollte, wenn Erwägungen dieser Art keine Rolle spielen, die Objektkomposition angewendet werden.
Betrachtet man den Ableitungsbaum von TORNE im Vergleich zu anderen mathematischen Frameworks wie Blitz++ (siehe [44]), so kann man sich die Frage stellen, warum z.B. die Matrix-Klassen nicht als parametrisierbare Typen implementiert wurden, die als Templates Bestandteil von C++ sind. Die Antwort hier ist eine ganz praktische: innerhalb der verwendeten Entwicklungsumgebung Visual-C++ 6.0 von Microsoft sind Templates instabil. Da Templates vom Compiler für die verschiedenen Typen entsprechend umgebaut werden müssen, hängt das Resultat stark von dieser Fähigkeit des Compilers ab. Mangels

Unterstützung wurde daher auf Templates verzichtet.
Ist die Compilerunterstützung besser, so steht einer Verwendung von Templates nichts im Wege.
Fazit: Das Resultat des Projektes TORNE ist ein Klassenframework, d.h. "eine Menge von Klassen, die einen wiederverwendbaren Entwurf für eine bestimmte Klasse von Software darstellen" [10]. Dies bedeutet für die Entwicklung von Programmen, dass natürlich nicht alle Klassen verwendet werden, die im Framework vorkommen, sondern nur die, welche für ein bestimmtes Verfahren notwendig sind.

2.3.3 Beispielnetzwerk und Ablaufkontrolle

Nachdem nun die grundlegenden Muster definiert sind, soll anhand eines Beispielsettings die Ablaufkontrolle erläutert werden. Ein Beispielprozessnetz bestehe aus den Teilmustern, die der Abbildung 2.4 entnommen werden können.
Die Ausführung soll gemäss der Pfeile im Diagramm erfolgen. Dabei "besitzt" CGeometer die Klassen CController und COutlet; CController enthält seinerseits die Klassen CProvider, CFilter und CMap. In CGeometer wird über CController iteriert und die Ausgabe über COutlet getätigt. Als Option stehen den Berechnungsklassen CSupport-Klassen zur Verfügung.
Nun muss die Frage geklärt werden, ob eine explizite oder implizite Ablaufkontrolle implementiert wird.
Dazu sind folgende Definitionen notwendig:

Definition 2.3.2. Ein Prozessnetz heisst *demand-driven*, wenn es ausgeführt wird, sobald eine Ausgabe gefordert wird. Es heisst *event-driven*, wenn die Änderung eines Objektzustandes seine Ausführung erzwingt.

Definition 2.3.3. Unter *expliziter Ablaufkontrolle* versteht man die Steuerung jedes Objekts von einem zentralen Beobachter aus, d.h. dieser wendet nach Bedarf Methoden verschiedener Objekte an.

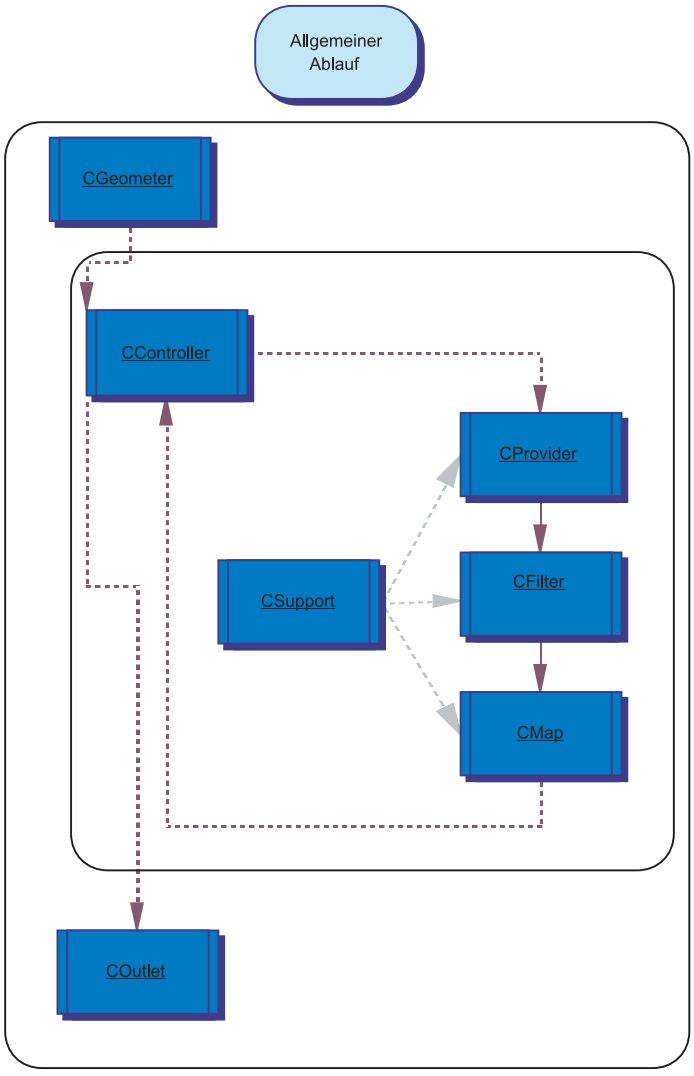

Abbildung 2.4: Beispielprozessnetz

Diese Form der Ablaufkontrolle kann sowohl event-driven, als auch demand-driven sein. Die Vorteile liegen darin, dass die Synchronisation und der Ablauf lokal für den Beobachter sind. Ausserdem ist eine leichtere Parallelisierung möglich. Nachteil ist die schwierige Kontrolle bei Ausführungsabhängigkeiten und die Tatsache, dass jedes Prozessobjekt vom Beobachter abhängig ist.

Definition 2.3.4. Bei *impliziter Ablaufkontrolle* führt ein Objekt sich selbst aus, wenn sich seine Parameter oder seine Eingaben ändern.

Die implizite Ablauflaufkontrolle ist demand-driven. Sie wird in einem Zweischrittverfahren implementiert (Update-Execute-Zyklus). Der Vorteil ist ihre Einfachheit: jedes Objekt beobachtet nur interne Zustände. Der Nachteil liegt in der schwierigeren Ausführungsoptimierung und Parallelisierung. Innerhalb eines Prozessnetzwerks wird bei letzterem Kontrollfluss lediglich der Teil ausgeführt, der sich ändert. Das ist im Visualisierungsbereich von Vorteil: man denke hier z.B. daran, dass ein gerendertes Objekt bei Änderung der Gesamthelligkeit nur neu beleuchtet, aber nicht ingesamt neu berechnet werden muss (siehe [39]). Innerhalb des Rekonstruktionsframeworks können ähnliche Prinzipien zum Tragen kommen. Beispiel hierfür ist die Möglichkeit, alle gefilterten Daten als Gesamtheit im Speicher zu halten und bei Änderungen der Auflösung des Voxel-Kubus nur noch die entsprechende Rückprojektion durchzuführen. Dass diese Option bisher noch nicht implementiert wurde liegt u.a. an den Resourcen. Trotzdem wurde für TORNE die implizite Ablaufkontrolle gewählt und zwar aus dem oben genannten Grund der Einfachheit.
Im nächsten Abschnitt wird der Update-Execute-Zyklus näher erläutert. Grundprinzip ist das folgende: Da alle Objekte mittels der Link-Methode an Parent und Child gebunden sind, wird über die Update-Methode ausgehend vom letzten Prozess der Kette (einem Outlet-Prozess) ein Signal an den Vorgänger weitergegeben. Dieser überprüft nun seinerseits bei dem ihm in der Ablaufkette vorgelagerten Prozess, ob dieser Up-to-date ist. Das geht solange weiter, bis die Kette am Anfang angekommen ist. Nun beginnt der Execute-Zyklus: jedes Objekt, dessen Vorgänger sich geändert hat, führt sich nun selbst aus und bringt sich somit auf den aktuellen Stand. Diese Änderung bedingt die Ausführung des

nachgelagerten Prozesses usw. Interessant dabei ist, dass die Abarbeitung erst dort beginnt, wo sich in der Kette zum ersten Mal ein Parameter ändert.

Beispiel 2.3.3. Als ein Diagramm kann man den Vorgang wie folgt darstellen:
[Abbildung 2.5]

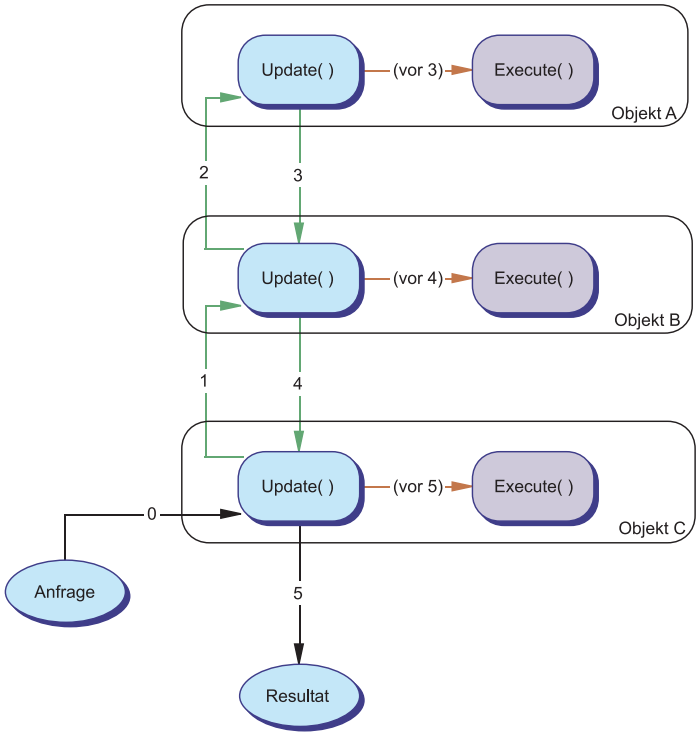

Abbildung 2.5: Update-Execute-Zyklus

In diesem Beispiel besteht das Prozessnetz aus Objekt A, B und C. Der Ablauf entspricht der Numerierung der Pfeile, wobei die "(vor x)"-Pfeile nur bei Notwendigkeit ausgeführt werden.

2.4 Rekonstruktionsalgorithmen

Im folgenden wenden wir uns nun der Implementierung der vorher besprochenen Rekonstruktionsverfahren mittels des Klassenframeworks TORNE zu. Dazu wird im einzelnen eine Ablaufstruktur in Metasprache gegeben und die resultierende Objektkomposition von Klassen des Musterkatalogs 3.1 aufgezeigt.

2.4.1 Approximative Inverse

Der Grundalgorithmus für die 3D-Rekonstruktion mittels Approximativer Inverse basiert auf der gefilterten Rückprojektion. Seine Darstellung in Metasprache ist wie folgt:

```
Setup der Daten
Für alle Projektionen:
    Ermittlung der Detektorkoordinaten
    Erzeugung oder Bereitstellung der Daten
    Fourier-Transformation der Daten
    Wenn nötig: Generierung des Filters
    Anwendung des Filters im Fourierbereich
    Inverse Fourier-Transformation der Daten
    gewichtete Rückprojektion
Ausgabe des Voxel-Kubus
```

Die Filtergenerierung kann unter Ausnutzung von Approximationen und Invarianzen auf eine eine einzige Projektion beschränkt werden. Deshalb ist eine Berechnung nur einmal notwendig. Es ergibt sich bei der Implementierung das Objektmodell laut Diagramm 2.6.

Das dynamische Modell zeigt die Gestalt von Abbildung 2.7.

Das Funktionalmodell ergibt sich dann gemäss Abbildung 2.8.

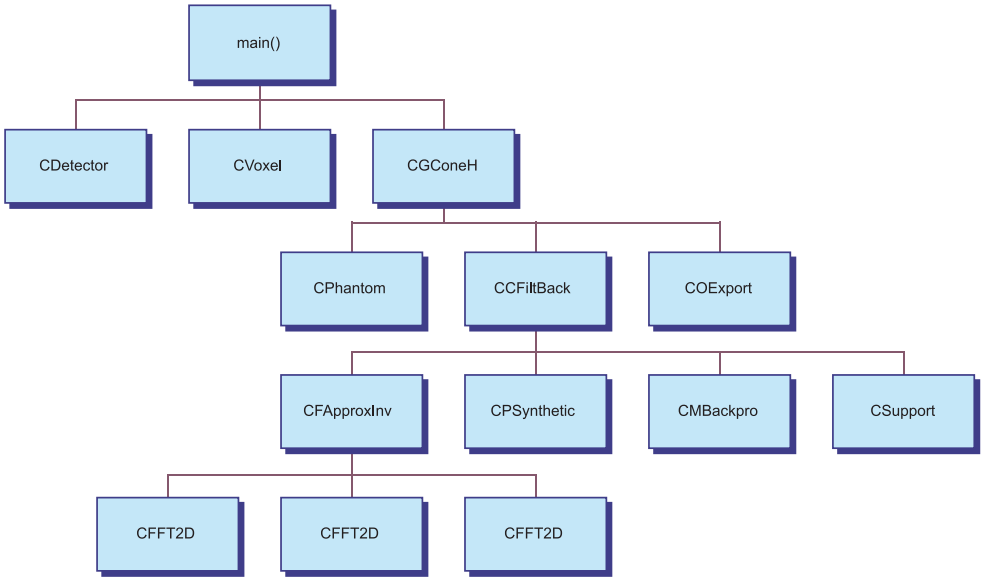

Abbildung 2.6: AI Objektmodell

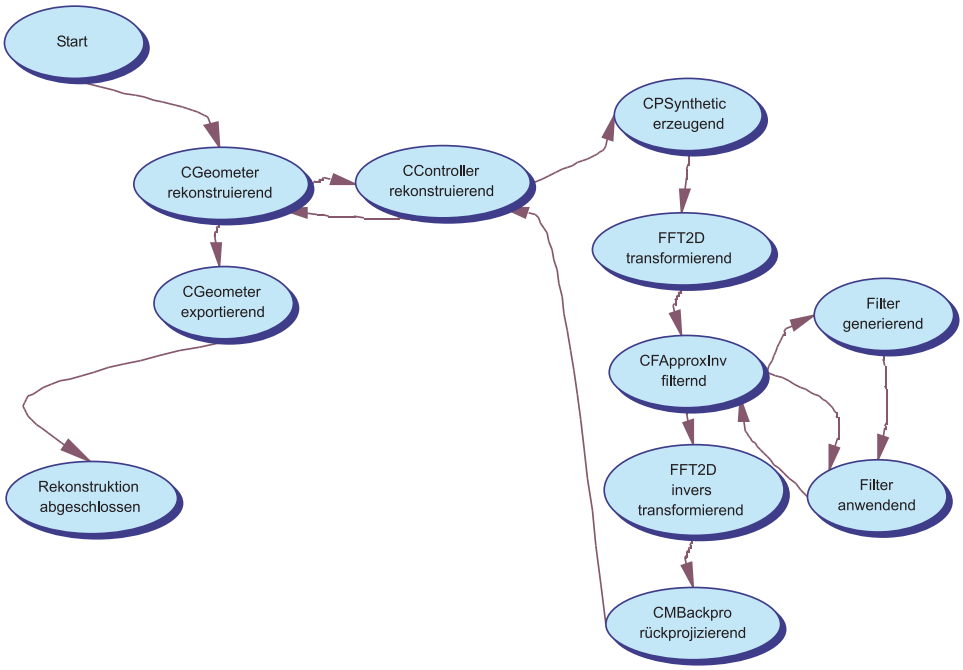

Abbildung 2.7: AI Dynamikmodell

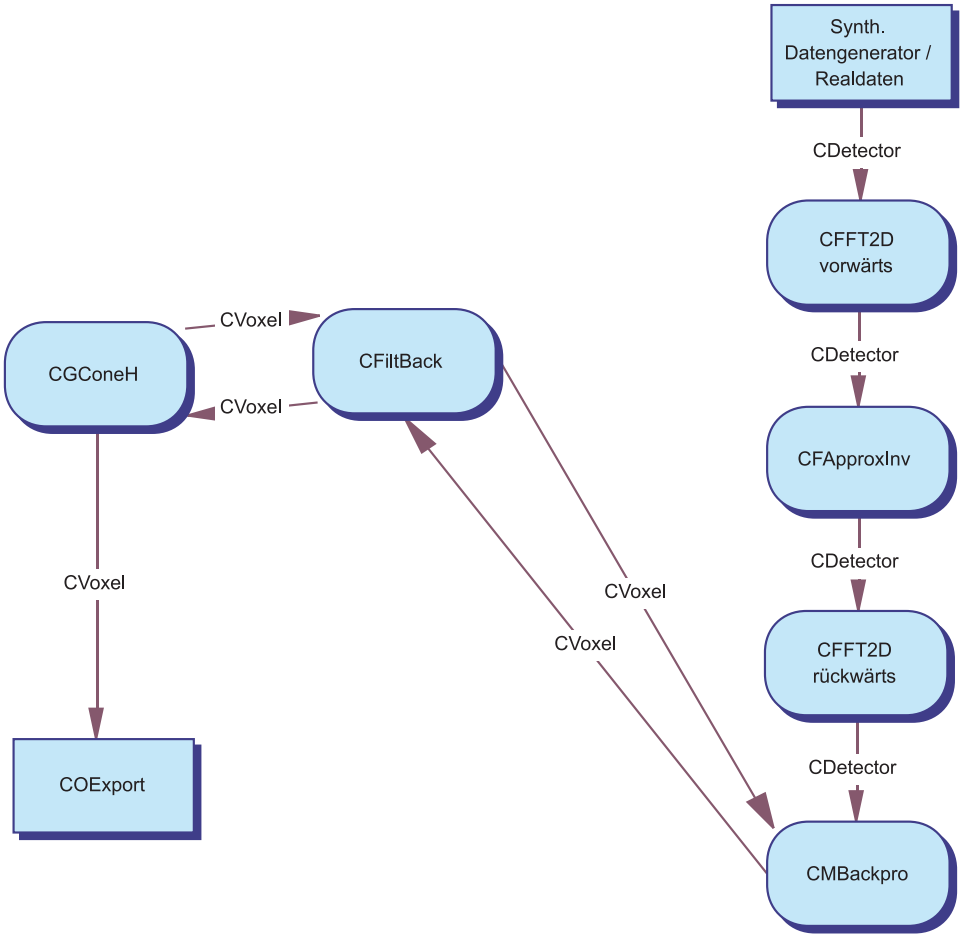

Abbildung 2.8: AI Funktionalmodell

2.4.2 Tomosynthese

Wie die Approximative Inverse basiert die Tomosynthese auf dem Verfahren der gefilterten Rückprojektion. Sie wird hier dargestellt, um zu zeigen, dass die tatsächlichen Änderungen am Programm für ein anderes Rekonstruktionsverfahren minimal sind:

```
Setup der Datenstrukturen
einmalige Generierung des Filters (richtungsunabhängig)
Für alle Projektionen:
    Ermittlung der Detektorkoordinaten
    Erzeugung oder Bereitstellung der Daten
    Fourier-Transformation der Daten
    Anwendung des Filters im Fourierbereich
    Inverse Fourier-Transformation der Daten
    ungewichtete Rückprojektion
Ausgabe des Voxel-Kubus
```

Der Filter ist richtungsunabhängig und muss daher nur einmal erzeugt werden. Es ergibt sich bei der Implementierung das Objektmodell der Abbildung 2.9.

Das dynamische Modell zeigt die Gestalt der Abbildung 2.10

Das Funktionalmodell ergibt sich gemäss Abbildung 2.11

Hier ist gut ersichtlich, welchen Vorteil die Implementierung mittels eines OOP-Stiles bringt: es werden wirklich nur die notwendigen Teile der Implementation modifiziert, die wiederverwendbaren Objekte müssen nicht neu geschrieben werden. Möglich wird dies dadurch, dass die Schnittstellen durch ihre Parametrisierung weitgehend unabhängig von Geometrie und Rekonstruktionsverfahren sind.

2.4.3 GUF-Verfahren

Als etwas komplexer in der Ablaufstruktur erweist sich das Verfahren der geometrieadaptiven Filterung. Während bei der gefilterten Rückprojektion der Ablauf für jede Projektion

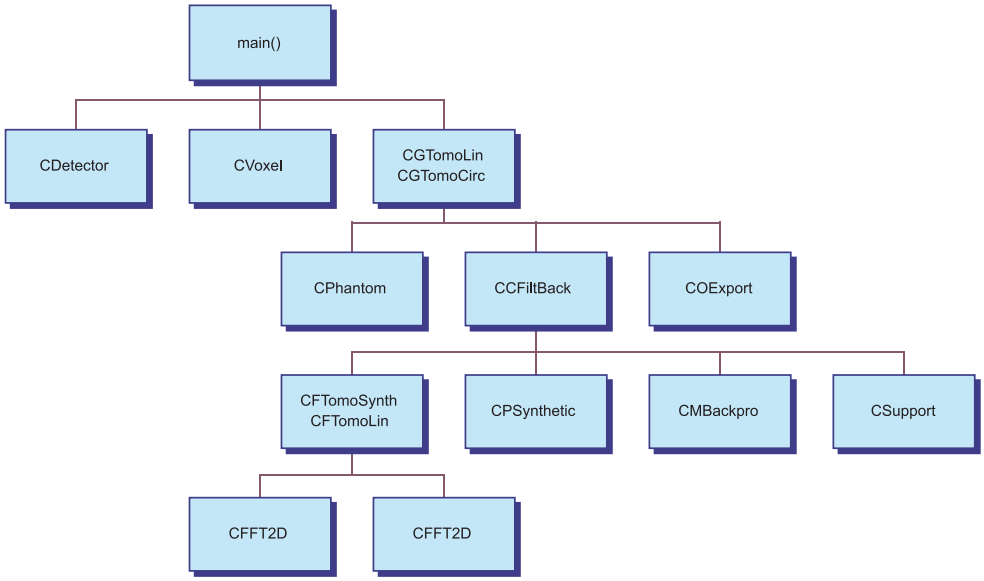

Abbildung 2.9: Tomosynthese Objektmodell

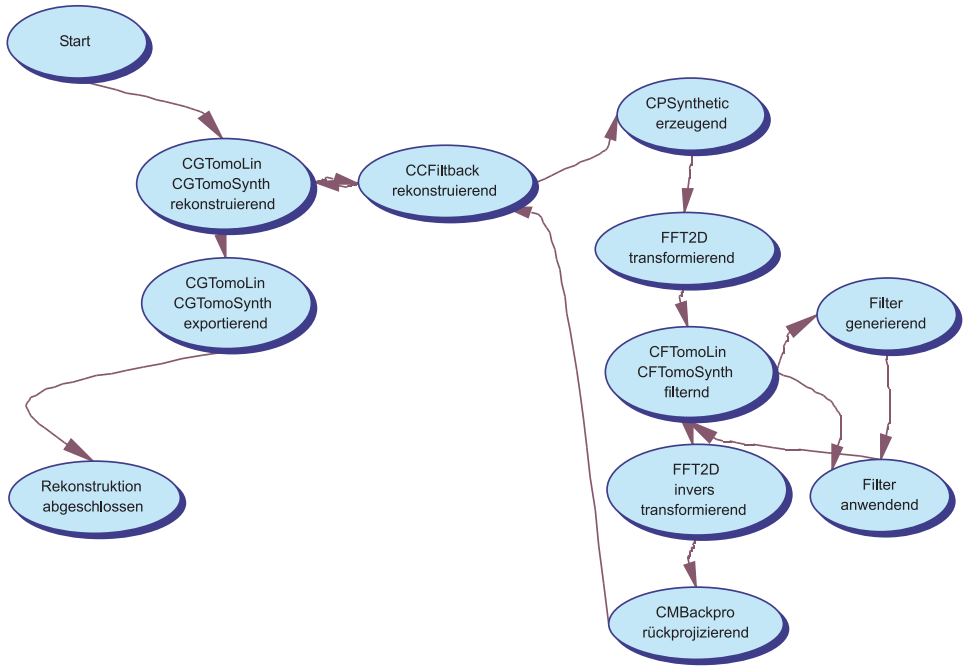

Abbildung 2.10: Tomosynthese Dynamikmodell

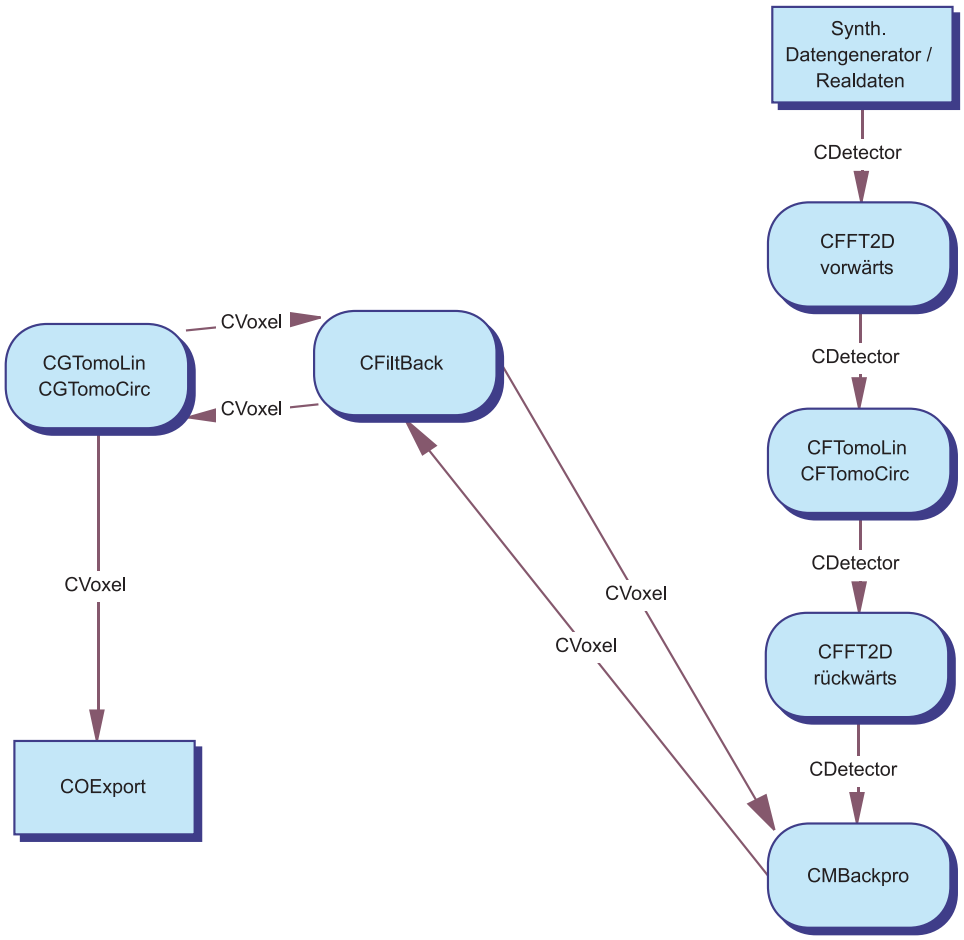

Abbildung 2.11: Tomosynthese Funktionalmodell

der gleiche ist und das Rekonstruktionsergebnis nach Bearbeitung der letzten Projektion direkt exportiert werden kann, sind in diesem Verfahren noch Operationen auf dem Voxel-Kubus notwendig.
Es ergeben sich für eine Rekonstruktion folgende Schritte:

```
Setup der Datenstrukturen
Controller-Setup
Für alle Projektionen:
    Ermittlung der Detektorkoordinaten
    Erzeugung oder Bereitstellung der Daten
    Rückprojektion
    Projektion des Basiselementes auf virtuellen Detektor
    Rückprojektion des Basiselementes
Berechnung des exakten Basiselementes
3D-Fourier-Transformation der rückprojizierten Basis
3D-Fourier-Transformation der exakten Basis
Filter-Generierung (komplexe Division)
Regularisierung (Gauss-Fenster, Nullfrequenzerhaltung)
3D-Fourier-Transformation der rückprojizierten Daten
Anwendung des Filters im Fourier-Bereich
inverse 3D-Transformation der Daten
Ausgabe des Voxel-Kubus
```

Die entsprechende Umsetzung im Klassenframework erfordert dementsprechend mehr Objekte. Das Objektmodell ist wie in Abbildung 2.12 angegeben.
Das dynamische Modell ist in Abbildung 2.13 dargestellt.
Ein vereinfachtes Funktionalmodell kann Abbildung 2.14 entnommen werden.

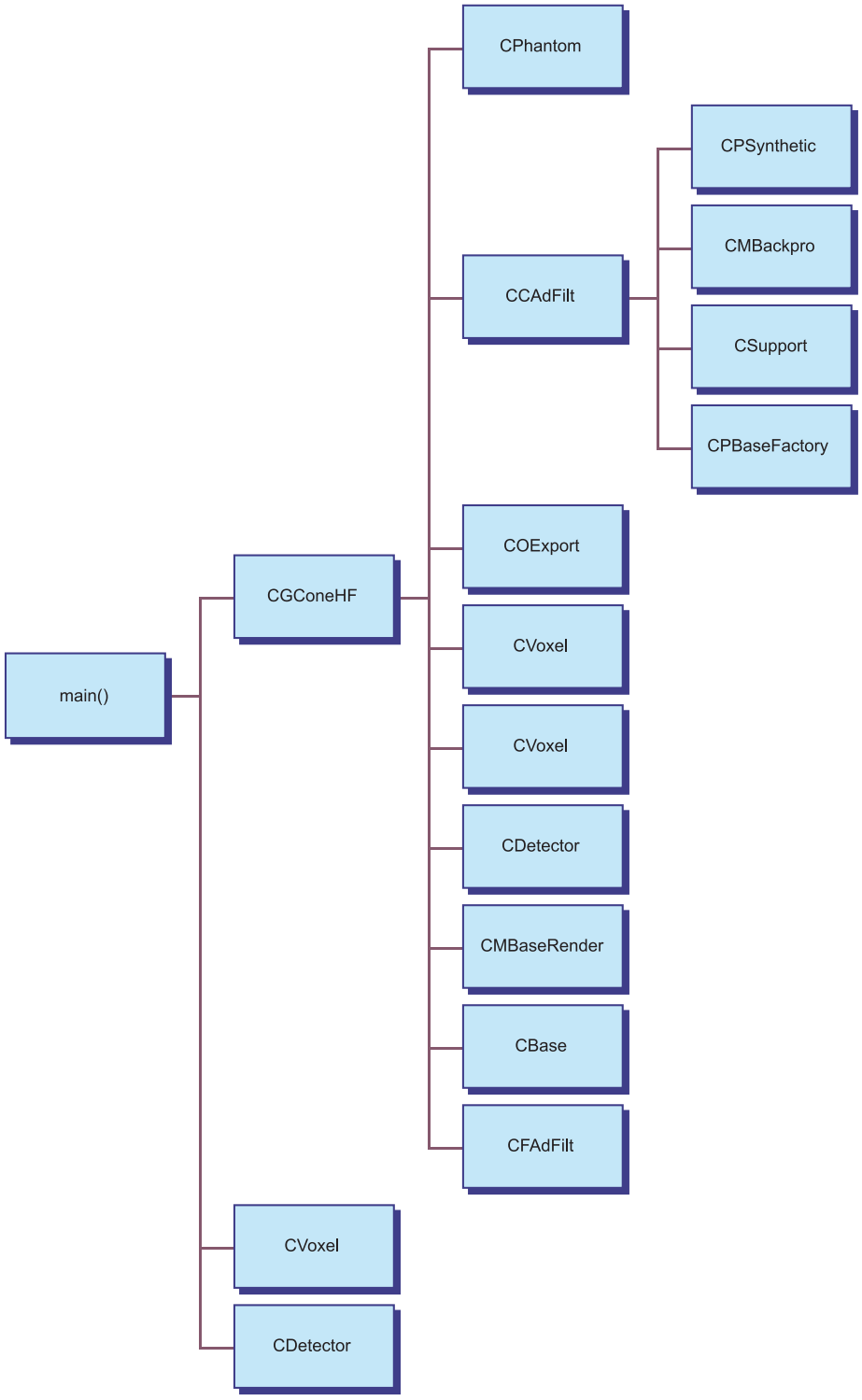

Abbildung 2.12: GUF Objektmodell

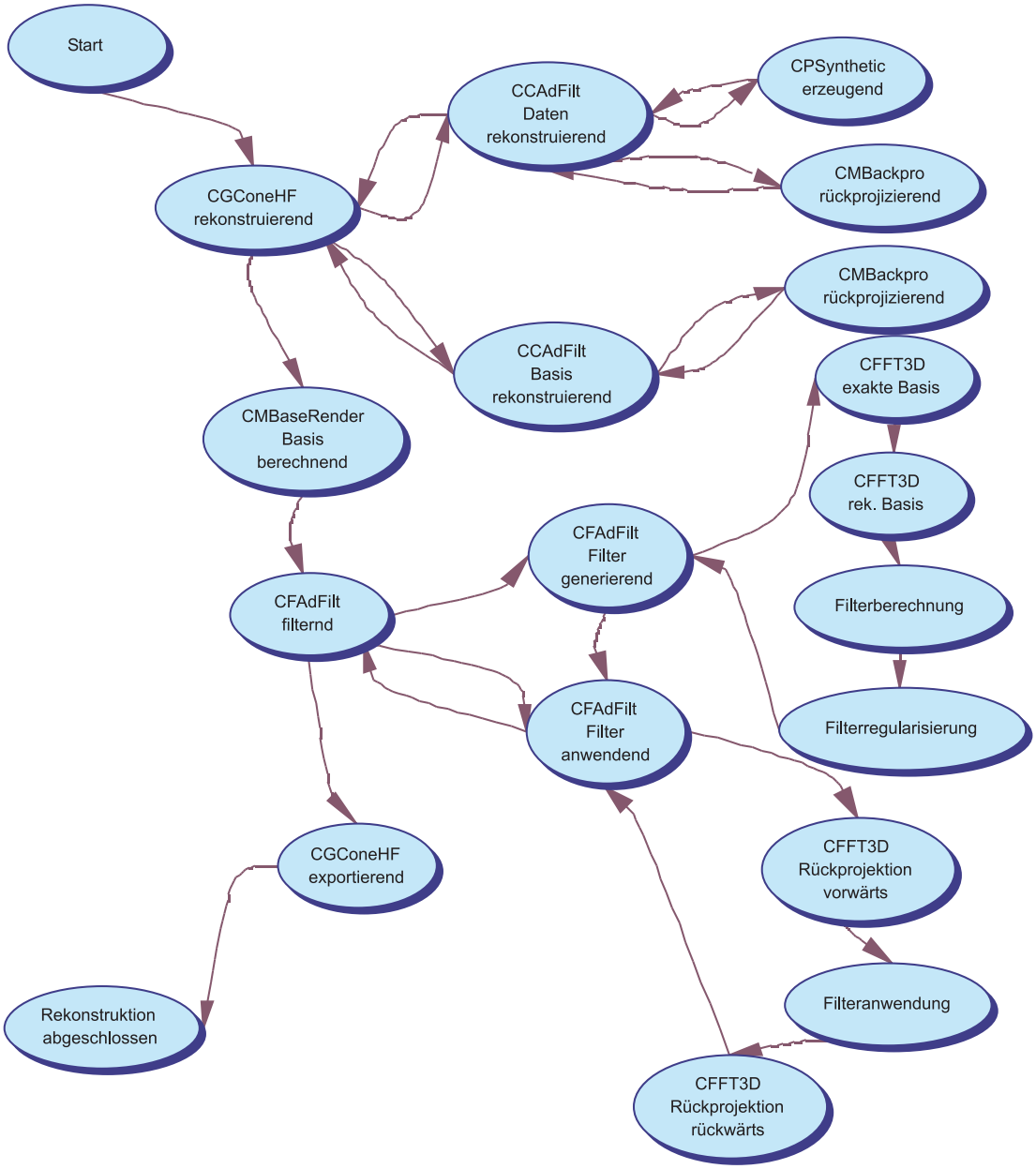

Abbildung 2.13: GUF Dynamikmodell

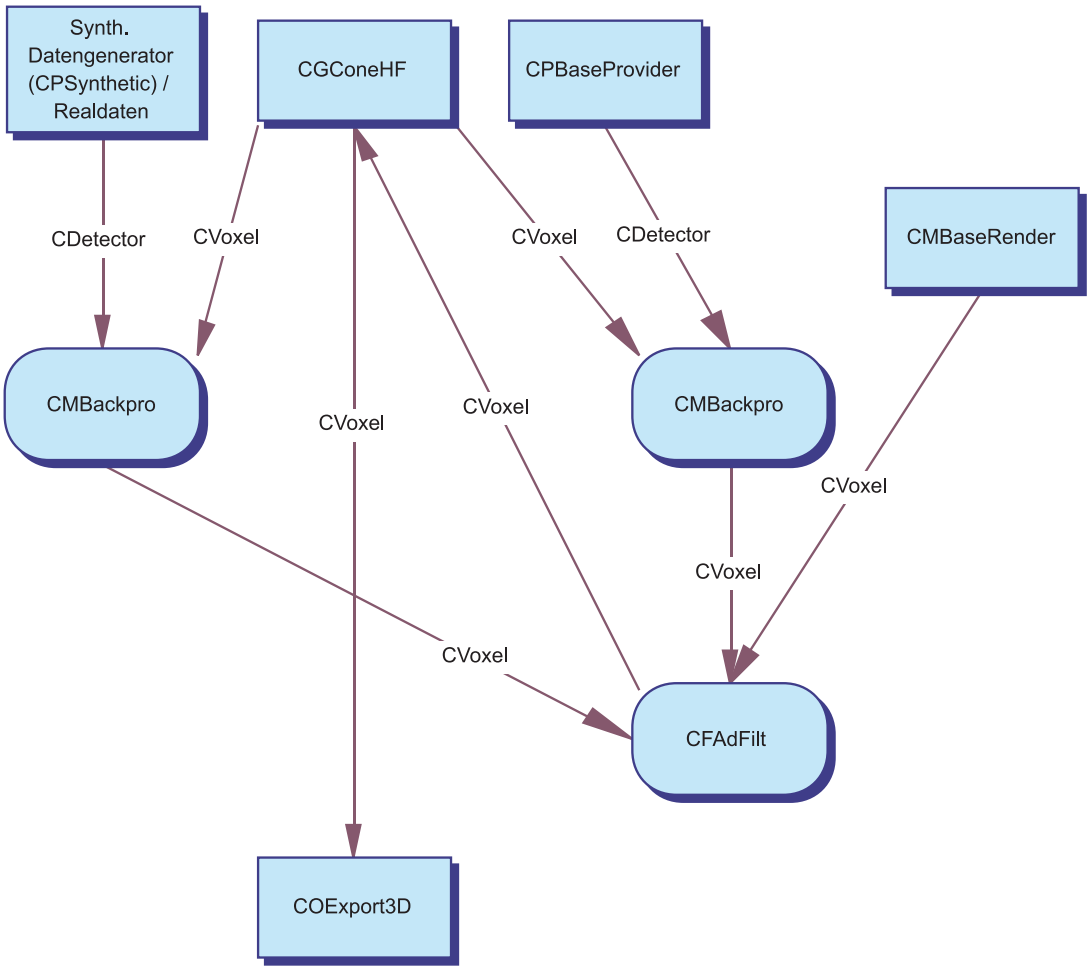

Abbildung 2.14: GUF Funktionalmodell

Kapitel 3

Objekt- und Musterkatalog

3.1 Übersicht

In diesem Abschnitt wird eine komplette Auflistung des aktuellen Muster- und Klassenbestandes von TORNE angegeben. Die Abbildungen 3.1 bis 3.4 zeigen die Objektmodelldiagramme der entsprechenden Teile.

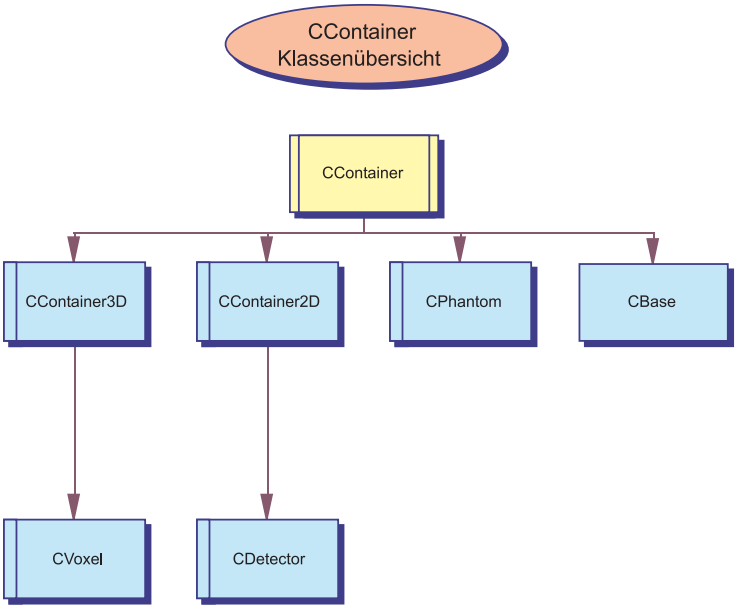

Abbildung 3.1: Container

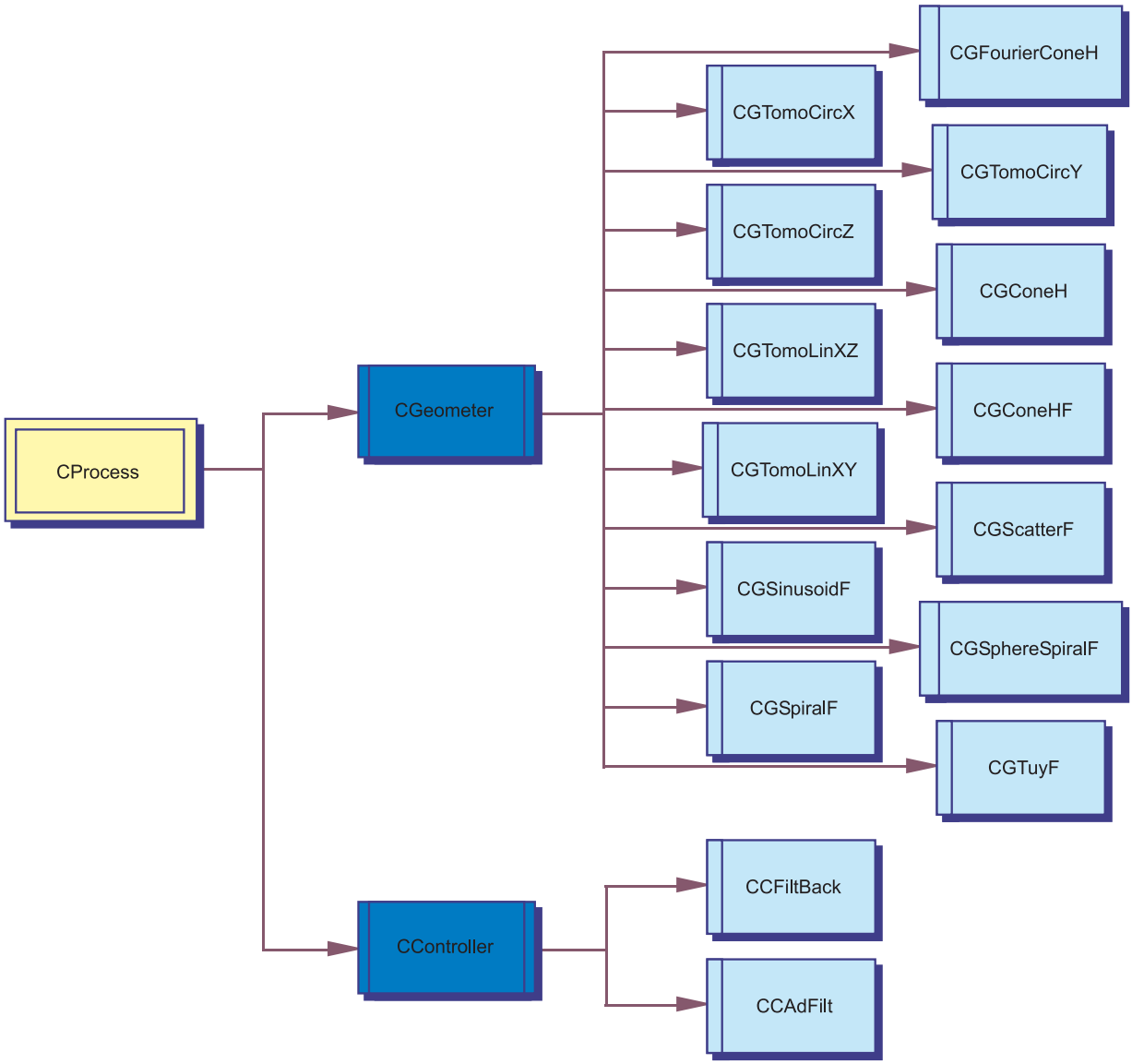

Abbildung 3.2: Geometer und Controller

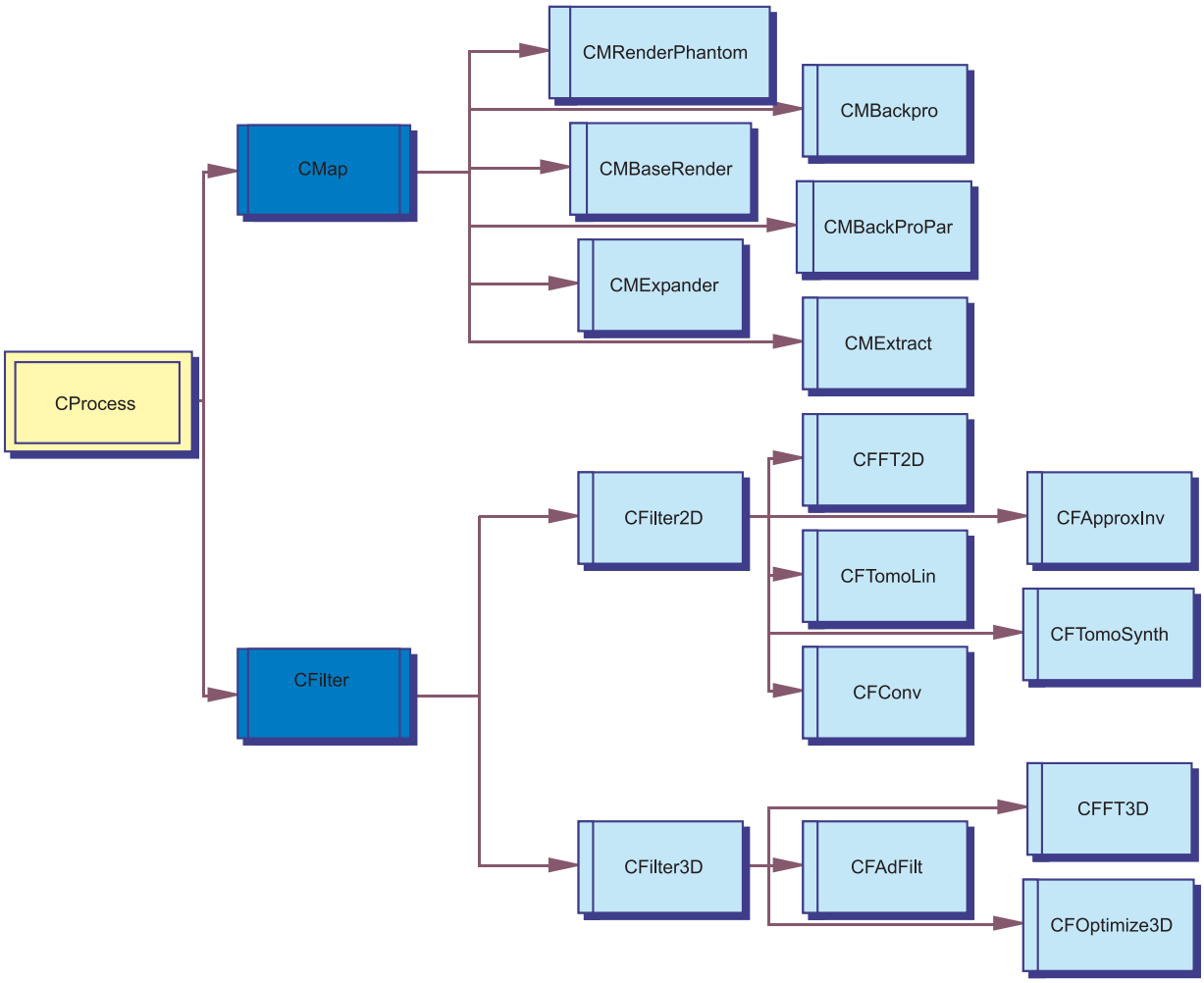

Abbildung 3.3: Filter und Mapper

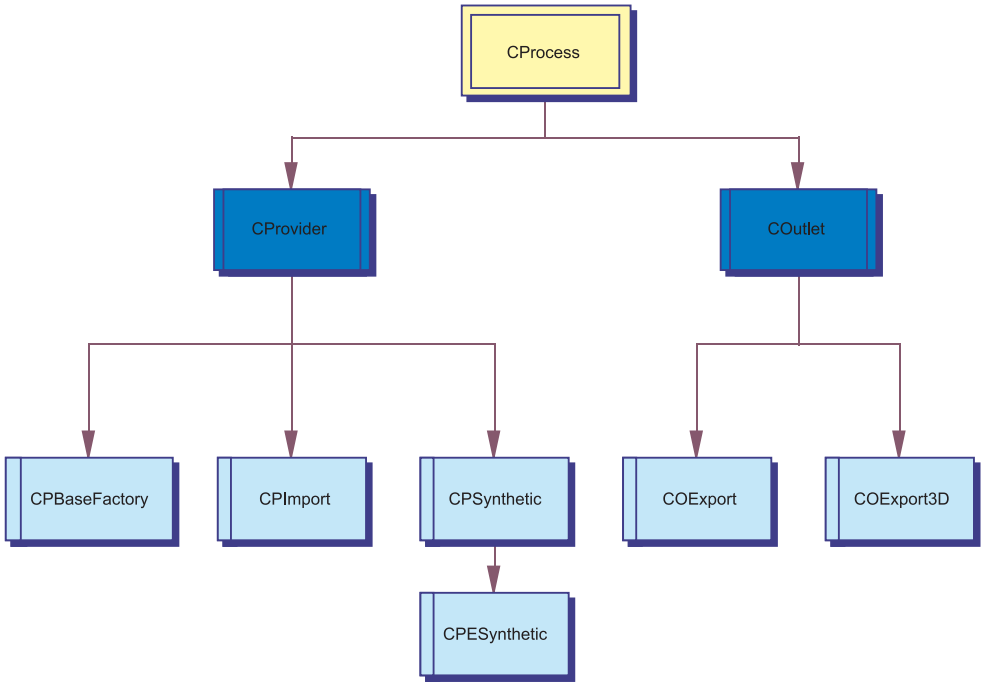

Abbildung 3.4: Provider und Outlet

In den Tabellen des Musterkataloges sind die Objekte wie folgt beschrieben:

- Funktion: nennt kurz die Funktion innerhalb des Frameworks
- Muster: Benennung des Musters, unter das die Klasse fällt.
- Basisklasse: falls es sich um eine abgeleitete Klasse handelt, dann wird die Basisklasse genannt
- Methoden: eine Nennung aller Methoden, die von der Klasse entweder überschrieben oder neu eingeführt werden. Die Methode 'Execute' wird von Prozessen generell überschrieben. Sind die Methoden nicht näher spezifiziert, so handelt es sich um public-Methoden, also um Teile der Schnittstelle. Auf private Methoden wird durch das Kürzel "(priv)" hingewiesen, protected Methoden sind durch "(prot)" gekennzeichnet.

- Attribute: Nennung der Klassenattribute. Für Kürzel gilt das unter 'Methoden' genannte.
- Optionen: werden (aus Effizienzgründen) mehrere Möglichkeiten der Ausführung geboten (welche prinzipiell als eigene Klasse implementiert werden könnten), so werden sie unter diesem Punkt aufgelistet.
- *Bemerkung*: hier wird bei Bedarf kurz auf die mathematischen bzw. informatischen Aspekte der Implementation eingegangen.

Bemerkung 3.1.1 (Quellcodestatistik). Das Projekt TORNE umfasst in der aktuellen Version 52 Klassen und über 10000 Zeilen Sourcecode, wovon 2000 auf Kommentare und 8000 auf eigentlichen Code entfallen.

Bemerkung 3.1.2. Der Musterkatalog ist alphabetisch geordnet bis auf eine Ausnahme: Die Klassen werden zuerst genannt, die direkt von den mathematischen Mustern abgeleitet sind und abstrakte Basisklassen darstellen.

3.2 Katalog

3.2.1 Containerklassen

CContainer	
Funktion	abstrakte Basisklasse für alle Container
Muster	Container
Basisklasse	keine
Methoden	Setup - Erzeugung der Datenstrukturen Get - Lesezugriff auf Datenstrukturen Put - Schreibzugriff auf Datenstrukturen
Attribute	data (Prot) - Zeiger auf Datenelemente changed - Modifikationsflag

Bemerkung: Diese Klasse stellt eine reine Schnittstellendefinition dar. Alle Methoden sind virtuell.

CBase	
Funktion	Kapselung der analytischen Basiselementfunktionen
Muster	Container
Basisklasse	CContainer
Methoden	SetType - Legt den Elementtyp fest Get - Liefert den Funktionswert eines Basiselements GetRadon - Liefert die Radon-Transformation
Attribute	Base - Basiselementtyp (Radial, Square, Hat, Gauss) Basesize - Basiselementgrösse

Bemerkung: Dieser Container wertet entweder die entsprechende Basisfunktion an einer gewünschten Stelle aus oder berechnet analytisch die Röntgen-Transformation des Basiselements, also das Integral über den Schnitt des Elementes mit einer beliebigen Gerade im Raum.

CContainer2D	
Funktion	Basisklasse für alle matrixäquivalenten Container mit Dimension 2
Muster	Container
Basisklasse	CContainer
Methoden	Copy - Methode zum Kopieren von Daten in aktuellen Container Get - zweidimensionaler Lesezugriff Put - zweidimensionaler Schreibzugriff Resetup - Methode zur Dimensions- / Datentypmodikation
Attribute	isCmplx (Prot) - Flag für komplexe Daten

Bemerkung: Der grundlegende 2D-Container. Standardmässig ist das Datenformat double, unterstützt werden auch komplexe Zahlen.

CContainer3D	
Funktion	Basisklasse für alle matrixäquivalenten Container mit Dimension 3
Muster	Container
Basisklasse	CContainer
Methoden	Copy - Methode zum Kopieren von Daten in aktuellen Container Get - dreidimensionaler Lesezugriff Put - dreidimensionaler Schreibzugriff Resetup - Methode zur Dimensions- / Datentypmodikation
Attribute	isCmplx (Prot) - Flag für komplexe Daten iorigin - Integer-Nullpunkt

Bemerkung: Analog zu CContainer2D implementiert diese Klasse den grundlegenden 3D-Container. Auch hier ist standardmässig das Datenformat double, unterstützt werden ebenfalls komplexe Zahlen.

CDetector	
Funktion	Mathematische Modellierung eines zweidimensionalen Detektors
Muster	Container
Basisklasse	CContainer2D
Methoden	GetC - Lesezugriff mit Realkoordinaten PutC - Schreibzugriff mit Realkoordinaten CopyPar - Parameterübernahme eines anderen Detectors
Attribute	D - Abstand Zentrum-Detektor R - Abstand Zentrum-Quelle del_x, del_y - Detektorelementgrösse det_x, det_y - Detektorgrösse middle_x, middle_y - Detektormitte disx, disy - Displacement phi, theta - Lage des Detektors über Kugelkoordinaten phid, thetad - Verkippung des Detektors über Kugelkoordinaten a - Quellposition (3D-Koordinaten) direc - Richtungsvektor zum Ursprung ortho - Richtungsvektor senkrecht Detektor

Bemerkung: Der Detektor ist so modelliert, dass er Parameter für eine beliebige Lage im Raum enthält, also mit variabler Position und Verkippung. Die Parametrisierung erfolgt über Kugelkoordinaten oder Angabe von Richtungsvektoren.

CPhantom	
Funktion	Kapselung des mathematischen Phantoms
Muster	Container
Basisklasse	CContainer
Methoden	SetFilename, LoadPhantom - Phantomdaten einlesen Get - speziell parametrisierte Zugriffsroutine
Attribute	origin - Zentrum scale - Skalierung

Bemerkung: Die hier unterstützten mathematischen Phantome bestehen aus einer Reihe von Ellipsoiden mit bestimmten Dichtewerten, die frei positioniert und gedreht sein können. Wird keine Phantombeschreibung geladen, so wird standardmässig das Shepp-Logan-Phantom verwendet.

CVoxel	
Funktion	Modellierung eines dreidimensionalen Voxel-Kubus
Muster	Container
Basisklasse	CContainer3D
Methoden	GetC - Lesezugriff mit Realkoordinaten PutC - Schreibzugriff mit Realkoordinaten
Attribute	theta, phi - Verkippung parametrisiert über Kugelkoordinaten vox_x, vox_y, vox_z - Voxelkubusgrösse del_x, del_y, del_z - Voxelelementgrösse origin - Nullpunkt

Bemerkung: Der Voxelkubus ist, wie der Detektor, frei im Raum zu positionieren und zu verkippen. Das Innere stellt jedoch immer eine äquidistant und kantenparallel diskretisierte Punktmenge dar.

3.2.2 Prozessklassen

CProcess	
Funktion	Basisklasse für alle Prozessobjekte
Muster	Process
Basisklasse	keine
Input	CContainer
Output	CContainer
Methoden	SetDName, DOut - Ausgabe der Prozessdaten zu Debuggingzwecken Link - Verbindung zweier Prozess (Ablauf und Datenstrom) SetChild, SetParent - Einzelverbindung zweier Prozesse bzgl. Ablauf SetInput, SetOutput - Einzelverbindung zweier Prozesse bzgl. Datenstrom GetInput, GetOutput - Zugriff auf Ein- und Ausgabedaten Update, Execute - Ausführungsschleife des aktuellen Prozesses (siehe 2.5) Report - Statusreport des Prozesses
Attribute	keine öffentlichen

Bemerkung: Als Basisklasse aller Prozessobjekte besitzt CProcess die notwendige Funktionalität, um ein Prozessnetzwerk aufzubauen. Weiterhin ist die Grundlage der Statusausgabe gelegt und ein Identifikationsmechanismus vorhanden.

CController	
Funktion	Basisklasse für Rekonstruktionsalgorithmen
Muster	Controller
Basisklasse	CProcess
Input	CContainer
Output	CContainer
Methoden	keine
Attribute	actno - Nummer des aktuellen Rekonstruktionsschritts

Bemerkung: Ausser des zusätzlichen Attributs wird die Schnittstelle nicht erweitert.

CFilter	
Funktion	Basisklasse für Filterungsprozesse
Muster	Filter
Basisklasse	CProcess
Input	CContainer
Output	CContainer
Methoden	Generate - Filterberechnung Apply - Anwendung des Filters
Attribute	filter - zu verwendender Filter

Bemerkung: Die Schnittstellendefinition der Filterklassen.

CGeometer	
Funktion	Basisklasse für Geometriegeneratoren
Muster	Geometer
Basisklasse	CProcess
Input	CContainer
Output	CContainer
Methoden	ReportStep - Statusreport für die aktuelle Projektion
Attribute	NProj - Anzahl Projektionen

Bemerkung: Als Gemeinsamkeit der Geometriegeneratoren tritt die Anzahl der Projektionen und die Notwendigkeit des Statusreports bei der aktuellen Projektion auf.

CMap	
Funktion	Basisklasse für Mapper-Objekte
Muster	Mapper
Input	CContainer
Output	CContainer
Basisklasse	CProcess

Bemerkung: Die abstrakte Basisklasse für alle Mapper-Objekte.

COutlet	
Funktion	Basisklasse für alle Ausgaben
Muster	Outlet
Basisklasse	CProcess
Input	CContainer
Output	CContainer (optional)

Bemerkung: Abstrakte Basisklasse für alle Ausgabe-Objekte.

CProvider	
Funktion	Basisklasse für alle Eingabeprozesse
Muster	Provider
Basisklasse	CProcess
Input	CContainer
Output	CContainer

Bemerkung: Abstrakte Basisklasse für alle Provider-Objekte.

CCFiltBack	
Funktion	Controller für gefilterte Rückprojektion
Muster	Controller
Basisklasse	CController
Input	CDetector
Output	CVoxel
Attribute	phi0 - Startwinkel R - Abstand Quelle - Fokus D - Abstand Quelle - Detektor Nproj - Anzahl Projektionen delphi - Winkeldiskretisierung fpar1, fpar2 - Filterparameter

Bemerkung: Der Grundablauf dieses Controllers besteht in Datenfilterung (evtl. Filtergenerierung, Transformation) und Rückprojektion.

CCAdFilt	
Funktion	Controller für das GUF-Verfahren
Muster	Controller
Basisklasse	CController
Input	CDetector
Output	CVoxel
Attribute	phi0 - Startwinkel R - Abstand Quelle - Fokus D - Abstand Quelle - Detektor Nproj - Anzahl Projektionen actphi - aktuelle Winkelposition base - assoziierter Basiscontainer

Bemerkung: Der Controller für das GUF-Verfahren kontrolliert die Rückprojektion von Daten und assoziiertem Basiselement.

CFilter2D	
Funktion	Basisklasse für 2D-Filter
Muster	Filter
Basisklasse	CFilter
Input	CContainer2D
Output	CContainer2D
Methoden	Apply - Anwendung des 2D-Filters
Attribute	

Bemerkung: Die Apply-Funktion verarbeitet jede Kombination von realen und komplexen Filtern und Daten mit optimierter Laufzeit.

CFApproxInv	
Funktion	Filter für Approximative Inverse
Muster	Filter
Basisklasse	CFilter2D
Input	CDetector
Output	CDetector
Methoden	Generate - Kernberechnung für die Approximative Inverse in 2D
Attribute	fpara1 - Diskretisierung der Winkelpositionen fpara2 - Gamma

Bemerkung: Dieser Filter ist speziell auf die Cone-Beam-Geometrie mit Abtastkurve in der XY-Ebene zugeschnitten. Die Filterberechnung ist laufzeitoptimiert und der Filter wird, einmal berechnet, auf einem sekundären Speichermedium gesichert und wiederverwertet.

CFConv	
Funktion	Implementation eines allgemeinen Faltungsfilters
Muster	Filter
Basisklasse	CFilter2D
Input	CDetector
Output	CDetector
Methoden	conv - Faltung (P) scan - Optimierung (P)
Attribute	byfft - Wahl der Faltung

Bemerkung: Der allgemeine Faltungsfilter faltet die Eingabe mit der angebundenen Filtermatrix, welche nicht von dieser Klasse berechnet wird. Dies geschieht entweder unter Ausnutzung des Faltungssatzes und FFT oder durch eine Ortsraumfaltung mit vorheriger Optimierung, bei der der Träger des Filters ermittelt wird.

CFFT2D	
Funktion	zweidimensionale Fourier-Transformation
Muster	Filter
Basisklasse	CFilter2D
Input	CContainer2D
Output	CContainer2D
Methoden	Apply - Schnittstelle zur Fourier-Transformation Generate - Schnittstelle zur Optimierung Init2D - Optimierung der Transformation Shift - Fourier-Shifting (Vertauschung der Quadranten) Transform2D - eigentliche Anwendung der Fourier-Transformation
Attribute	Direction - Richtung der Transformation

Bemerkung: Diese Funktion basiert auf der FFTW-Library (siehe [9]), die durch Vorkalkulationen die Fourier-Transformation an die Datengrösse anpasst und optimiert.

CFTomoLin	
Funktion	Filter für lineare Tomosynthese
Muster	Filter
Basisklasse	CFilter2D
Input	CDetector
Output	CDetector
Methoden	Generate - Filtergenerierung
Attribute	para1 - Lowpass-Setting para2 - Highpass-Setting

Bemerkung: Der speziell auf die lineare Tomosynthese zugeschnittene Filter im Verfahren der gefilterten Rückprojektion.

CFTomoSynth	
Funktion	Filter für zirkulare Tomosynthese
Muster	Filter
Basisklasse	CFilter2D
Input	CDetector
Output	CDetector
Methoden	Generate - Filtergenerierung
Attribute	para1 - Lowpass-Setting para2 - Highpass-Setting

Bemerkung: Der spezielle Filter für die zirkulare Tomosynthese im Verfahren der gefilterten Rückprojektion.

CFilter3D	
Funktion	Basisklasse für 3D-Filter
Muster	Filter
Basisklasse	CFilter
Input	CContainer3D
Output	CContainer3D
Methoden	Apply - Anwendung des 3D-Filters
Attribute	

Bemerkung: Das Analogon zu CFilter2D in drei Dimensionen.

CFAdFilt	
Funktion	Filter für g-adaptive Filterung
Muster	Filter
Basisklasse	CFilter3D
Input	CVoxel
Output	CVoxel
Methoden	Execute - Transformation der Eingabedaten und Anwendung des Filters SetupFilter - Filterberechung aus Referenzdaten Regularize - spezielle Regularisierung
Attribute	cutfreq - Abschneidefrequenz cutwin - Fenster des Abschneidens dila - Trägerkonstante preserve - Durchschnittserhaltung
Optionen	Durchschnittserhaltung, Regularisierung

Bemerkung: Der geometrieadaptive Filter wird aus den Referenzdaten für die Basisfunktion berechnet und durch Anwendung eines dreidimensionalen Gauss-Fensters regularisiert. Es ist die Option vorhanden, mittels Nullpunkterhaltung den Durchschnittswert der Rekonstruktion beizubehalten.

CFFT3D	
Funktion	dreidimensionale Fourier-Transformation
Muster	Filter
Basisklasse	CFilter3D
Input	CContainer3D
Output	CContainer3D
Methoden	Apply - Schnittstelle zur Fourier-Transformation Generate - Schnittstelle zur Optimierung Init2D - Optimierung der Transformation Shift - Fourier-Shifting (Vertauschung der Quadranten) Transform2D - eigentliche Anwendung der Fourier-Transformation
Attribute	Direction - Richtung der Transformation ShiftBefore - Fourier-Shifting von der Transformation ShiftAfter - Fourier-Shifting nach der Transformation

Bemerkung: Wie CFFT2D setzt auch diese Klasse auf die FFTW-Library auf.

CFOptimize3D	
Funktion	Optimierung eines 3D-Containers zur Visualisierung
Muster	Filter
Basisklasse	CFilter3D
Input	CContainer3D
Output	CContainer3D
Methoden	Histo - Histogrammerstellung eines Voxel-Kubus
Attribute	UpperTresh - oberer Schwellwert LowerTresh - unterer Schwellwert

Bemerkung: Die Optimierung des Darstellungsbereiches eines Voxel-Kubus durch Abbildung auf 16-Bit Integerwerte. Dies erfolgt über Fenstersetzung mittels eines kumulativen Histogramms. Die Angaben der Schwellwerte sind in Prozent. Zwischen den minimal und maximal darzustellenden Funktionswerten wird eine lineare Zuteilung vorgenommen.

CGConeH	
Funktion	Geometriegenerator für die Cone-Beam-Geometrie (Horizontal)
Muster	Geometer
Basisklasse	CGeometer
Input	CDetector
Output	CVoxel
Attribute	phi0 - Startwinkel R - Abstand Quelle - Fokus D - Abstand Quelle - Detektor Nproj - Anzahl Projektionen delphi - Winkeldiskretisierung

Bemerkung: Die Eingabeparameter dieses Geometriegenerators decken sich grösstenteils mit dem entsprechenden Controller. In der Execute-Funktion berechnet er die Quell- und Detektorpositionen pro Winkel und führt den damit verbundenen Controller aus.

CGConeHF	
Funktion	Geometriegenerator für Cone-Beam-Geometrie (H) und GUF- Filter
Muster	Geometer
Basisklasse	CGeometer
Input	CDetector
Output	CVoxel
Attribute	phi0 - Startwinkel R - Abstand Quelle - Fokus D - Abstand Quelle - Detektor Nproj - Anzahl Projektionen delphi - Winkeldiskretisierung

Bemerkung: Auch dieser Controller berechnet Quell- und Detektorpositionen pro Winkel. Die Kernschleife besteht allerdings in Basiselement- und Datenrückprojektion mit nachfolgender 3D-Filterberechnung und Anwendung.

CGFourierConeH	
Funktion	Geometriegenerator für Fourier-Rekonstruktion, Cone-Beam-Geometrie
Muster	Geometer
Basisklasse	CGeometer
Input	CDetector
Output	CVoxel
Attribute	R - Abstand Quelle - Fokus D - Abstand Quelle - Detektor Nproj - Anzahl Projektionen delphi - Winkeldiskretisierung

Bemerkung: Dieser Geometer ist noch im experimentellen Stadium.

CGScatterF	
Funktion	Geometriegenerator für Monte-Carlo-Geometrie und GUF-Filter
Muster	Geometer
Basisklasse	CGeometer
Input	CDetector
Output	CVoxel
Attribute	R - Abstand Quelle - Fokus D - Abstand Quelle - Detektor Nproj - Anzahl Projektionen

Bemerkung: Der GUF-Geometriegenerator für die Monte-Carlo-Geometrie, siehe 1.4.2.1. Die Winkelpositionen werden über einen einfachen Zufallsgenerator randomisiert.

CGSinusoidF	
Funktion	Geometriegenerator für Sinusoidalgeometrie und GUF- Filter
Muster	Geometer
Basisklasse	CGeometer
Input	CDetector
Output	CVoxel
Attribute	R - Abstand Quelle - Fokus D - Abstand Quelle - Detektor Nproj - Anzahl Projektionen period - Anzahl Schwingungen

Bemerkung: Der GUF-Geometriegenerator für die Sinusoidalgeometrie. Er erlaubt eine freie Wahl der projizierten Sinus-Schwingungen. Siehe Abschnitt 1.4.2.2.

CGSphereSpiralF	
Funktion	Geometriegenerator für sphärische Spiralgeometrie und GUF- Filter
Muster	Geometer
Basisklasse	CGeometer
Input	CDetector
Output	CVoxel
Attribute	R - Abstand Quelle - Fokus D - Abstand Quelle - Detektor Nproj - Anzahl Projektionen Ncircles - Anzahl Umläufe

Bemerkung: Der Geometriegenerator für die sphärische Spiralgeometrie laut Abschnitt 1.4.2.2. Zusätzlicher Parameter ist die Anzahl der Umläufe für eine Abtastung mit $\theta \in [0, 2\pi]$.

CGSpiralF	
Funktion	Geometriegenerator für Spiral-Geometrie und GUF- Filter
Muster	Geometer
Basisklasse	CGeometer
Input	CDetector
Output	CVoxel
Attribute	R - Abstand Quelle - Fokus D - Abstand Quelle - Detektor Nproj - Anzahl Projektionen Ncircles - Anzahl Umläufe

Bemerkung: Auch bei der klassischen Spiralgeometrie ist der zusätzliche Parameter die Anzahl der Umläufe für eine Abtastung von h_{min} bis h_{max} des Zylinders.

CGTomoCircX, CGTomoCircY, CGTomoCircZ	
Funktion	Geometriegeneratoren für zirkulare Tomosynthese um X-,Y- oder Z-Achse
Muster	Geometer
Basisklasse	CGeometer
Input	CDetector
Output	CVoxel
Attribute	phi0, theta0 - Startwinkel R - Abstand Quelle - Fokus D - Abstand Quelle - Detektor Nproj - Anzahl Projektionen delphi - Winkeldiskretisierung tomoang - Tomowinkel

Bemerkung: Prinzipiell der Standard Cone-Beam-Geometrie ähnlich fährt bei dieser Geometrie die Quelle einen Kleinkreis um eine der Koordinatenachsen.

CGTomoLinXY, CGTomoLinYZ	
Funktion	Geometriegeneratoren für lineare Tomosynthese in XY (YZ)-Ebene
Muster	Geometer
Basisklasse	CGeometer
Input	CDetector
Output	CVoxel
Attribute	phi0, theta0 - Startwinkel R - Abstand Quelle - Fokus D - Abstand Quelle - Detektor Nproj - Anzahl Projektionen delphi - Winkeldiskretisierung tomoang - Tomowinkel detfix - Detektoroption
Optionen	stehender / bewegter Detektor

Bemerkung: Im Gegensatz zur zirkularen Tomosynthesegeometrie wird hier mit der Quelle ein Kreisbogen beschrieben. Der Detektor ist dabei entweder fixiert oder steht senkrecht zum Mittelstrahl.

CGTuyF	
Funktion	Geometriegenerator für die Standard-Tuy-Kurve und GUF- Filter
Muster	Geometer
Basisklasse	CGeometer
Input	CDetector
Output	CVoxel
Attribute	phi0 - Startwinkel R - Abstand Quelle - Fokus D - Abstand Quelle - Detektor Nproj - Anzahl Projektionen deltheta - Winkeldiskretisierung in θ delphi - Winkeldiskretisierung in ϕ

Bemerkung: Der Geometriegenerator für die klassische TUY–KIRILLOV–Anordnung zweier orthogonaler Kreise. Die Quellpositionen auf den Kreisen werden über *delphi* und *deltheta* parametrisiert.

CMBackpro	
Funktion	Cone-Beam-Rückprojektion
Muster	Mapper
Basisklasse	CMap
Input	CDetector
Output	CVoxel
Methoden	Backprojection - Standardrückprojektion BackprojectionW - gewichtete Rückprojektion BackprojectionF - Fourier-Rückprojektion
Attribute	Backprotype - Art der Rückprojektion
Optionen	Standard, gewichtet, Fourier

Bemerkung: Die Rückprojektionsklasse bietet für die Cone-Beam-Geometrie mehrere Arten der Rückprojektion, nämlich die ungewichtete (Standard-) Rückprojektion, die im Verfahren mittels Approximativer Inversen verwendete gewichtete Rückprojektion und die Fourier-Rückprojektion, die eine Übertragung von Informationen aus dem 2D-Spektrum in das 3D-Spektrum darstellt. Letztere ist noch im Teststadium. Alle Geometrieinformationen sind in den angebundenen Containerstrukturen enthalten.

CMBackproPar	
Funktion	parallele Rückprojektion
Muster	Mapper
Basisklasse	CMap
Input	CDetector
Output	CVoxel
Methoden	Backprojection - Standardrückprojektion BackprojectionF - Fourier-Rückprojektion
Attribute	Backprotype - Art der Rückprojektion
Optionen	Standard, Fourier

Bemerkung: Äquivalente Rückprojektionsklasse für die parallele Geometrie.

CMBaseRender	
Funktion	Basis-Renderer in Voxel-Kubus
Muster	Mapper
Basisklasse	CMap
Input	CBase
Output	CVoxel
Methoden	Execute - Rendert die Basis
Attribute	
Optionen	

Bemerkung: In seiner Funktion der Klasse CMRenderPhantom ähnlich, rendert diese Klasse Basiselemente in einen Voxelkubus. Die Parameter sind sämtlich in den Containern repräsentiert.

CMExpander	
Funktion	Voxel-Vergrösserung mit Interpolation
Muster	Mapper
Basisklasse	CMap
Input	CContainer3D
Output	CContainer3D
Methoden	Execute - expandiert den Voxel
Attribute	Interpolation - Wahl der Interpolation
Optionen	Nearest Neighbour, trilinear

Bemerkung: Dieser Mapper dient dazu, Voxelvolumen einer Dimension in eine andere abzubilden. Als Interpolationsformen bietet er Nearest-Neighbour-Interpolation und trilineare Interpolation. Die Grössenangaben werden aus den Dimensionen von Ein- und Ausgabecontainer ermittelt.

CMExtract	
Funktion	Slice-Extractor
Muster	Mapper
Basisklasse	CMap
Input	CContainer3D
Output	CContainer2D
Methoden	ParaSlice - parametrisierte Slice-Extraktion
Attribute	direction - Orientierung slice - logische Nummer des Slices

Bemerkung: Dieser Mapper gestattet eine achsenparallele Extraktion von zweidimensionalen Teilgebieten eines Voxel-Kubus (etwa zu Visualisationszwecken).

CMRenderPhantom	
Funktion	Phantom-Renderer in Voxel-Kubus
Muster	Mapper
Basisklasse	CMap
Input	CPhantom
Output	CVoxel
Methoden	Execute - Rendert das Phantom
Attribute	absval - absolute oder additive Dichten
Optionen	absolut, additiv

Bemerkung: Zur exakten Darstellung von Phantomdaten als Referenzoriginal rendert diese Klasse Phantome direkt in den Voxelkubus. Dies geschieht entweder durch Angabe absoluter Dichtewerte oder durch Addition der Dichtewerte der überschneidenden Ellipsoide.

COExport	
Funktion	einfache Exportklasse für 2D- und 3D-Daten
Muster	Outlet
Basisklasse	COutlet
Input	CContainer2D, CContainer3D
Output	= Input
Methoden	SetType - Angabe des Exporttyps SetName - Ausgabename
Attribute	etype
Optionen	short binary, double binary, short ascii, double ascii, float binary

Bemerkung: Je nach Eingabedatentyp wird eine zweidimensionale oder dreidimensionale Ausgabe vorgenommen. Für die zweidimensionale Ausgabe können sowohl Binär- als auch ASCII-Daten in float, short und double exportiert werden.

COExport3D	
Funktion	spezialisierte Ausgabe eines Voxel-Kubus im VTK-Format
Muster	Outlet
Basisklasse	COutlet
Input	CVoxel
Output	= Input
Methoden	SetName - Ausgabename festlegen
Attribute	upper - Oberer Schwellwert lower - unterer Schwellwert

Bemerkung: Unter Verwendung der Optimiererklasse wird eine gefensterte Version des Eingabekubus erstellt und diese im VTK-Format exportiert. Ist der Eingabekubus komplex, so werden Real- und Imaginärteil getrennt exportiert.

CPBaseFactory	
Funktion	Datengenerator für Basisfunktionen
Muster	Provider
Basisklasse	CProvider
Input	CBase
Output	CDetector
Methoden	GetRenBase - Berechnung der Strahlschnittpunkte
Attribute	basesize - Basiselementgrösse base - Basiselementtyp

Bemerkung: Der Basisprovider stellt mit Hilfe des analytischen Basiscontainers Linienintegrale über Basiselemente zur Verfügung. Es handelt sich hierbei also im Prinzip um einen speziellen Phantomgenerator. Er bildet mit CFAdFilt und CCAdFilt die Grundlage des GUF-Verfahrens.

CPESynthetic	
Funktion	verbesserter Generator für synthetische Daten
Muster	Provider
Basisklasse	CProvider
Input	CPhantom
Output	CDetector
Methoden	PreCalc - Vorberechnung der Rotationsmatrizen xray - Integral über einen Strahl datensynth - Bearbeitung aller Strahlen
Attribute	phantom - assoziiertes Phantom

Bemerkung: Der verbesserte Datengenerator (CP"Enhanced"Synthetic) ist in der Lage, Phantomdaten zu verarbeiten, bei denen den Ellipsoiden feste Dichtewerte zugewiesen wurden. Dies ermöglicht realistischere Phantome. Die Implementation verläuft über Schnittpunktberechnung von Ellipsen und Strahl, Sortierung, Eliminationsverfahren unter Verwendung eines Stacks und anschliessender Teilintegralberechnung.

CPImport	
Funktion	Import-Objekt für 2D-Daten
Muster	Provider
Basisklasse	CProvider
Input	CDetector (optional)
Output	CDetector
Methoden	SetType - Angabe des Exporttyps SetName - Eingabename
Attribute	etype
Optionen	short binary, double binary, short ascii, double ascii, float binary

Bemerkung: Analog zu COExport stellt diese Klasse die Möglichkeit zu Verfügung, eine Vielzahl von Eingabeformaten zu verarbeiten.

CPSynthetic	
Funktion	Datengenerator für synthetische Daten
Muster	Provider
Basisklasse	CProvider
Input	CPhantom
Output	CDetector
Methoden	xray - Integral über einen Strahl datensynth - Bearbeitung aller Strahlen
Attribute	phantom - assoziiertes Phantom

Bemerkung: Diese Providerklasse berechnet die Daten für ein gegebenes Phantom mittels Addition der Linienintegrale durch die Ellipsoide.

CSupport	
Funktion	Supportklasse
Muster	Process
Basisklasse	CProcess
Input	CContainer (redundant)
Output	CContainer (redundant)
Methoden	scale01 - Skalierung eines matrixäquivalenten Containers MatVec - Matrix / Vektor Multiplikation MatMat - Matrix / Matrix Multiplikation distance - Distanzberechnung in 3D topolar - Umrechnung in Polar- bzw. Kugelkoordinaten frompolar - Umrechnung zu Polar- bzw. Kugelkoordinaten GetAverage - Durchschnittsberechnung sgn - Signumsberechnung psgn - spezielle Signumsberechnung scalar - Skalarprodukt direction - orthogonale Richtungsvektoren
Attribute	keine

Bemerkung: Die in 2.3.2 beschriebene Supportklasse. Die enthaltenen Funktionen sind grösstenteils einfache Berechnungsroutinen für Vektoren und Skalare.

Anhang A

Das verwendete Phantom

Dieses Phantom erlaubt die Untersuchung von Auflösung bezüglich Dichtesprüngen und Ellipsoidgrössen. Die inneren Ellipsoide entsprechen verschiedenen Gewebe- und Materialdichten, insbesondere denen von Zahnschmelz, Dentin, Knochen und Weichgewebe bezüglich einer Strahlung von 60kV (Kennwerte entnommen aus [13] und [15]).

Nummer	rel. Dichte	abs. Dichte	mx	my	mz	rx	ry	rz
1	1	2	0	0	0.75	0.1	0.1	0.15
2	1	2	0	0	0.4	0.05	0.05	0.1
3	1	2	0	0	0.15	0.03	0.03	0.05
4	1	2	0	0	0	0.01	0.01	0.02
5	1	2	0	0	-0.15	0.03	0.03	0.05
6	1	2	0	0	-0.4	0.05	0.05	0.1
7	1	2	0	0	-0.75	0.1	0.1	0.15
8	1	1	0	0	0	1	1	1
9	0.1	1.1	0.75	0	0	0.15	0.15	0.15
10	0.2	1.2	0.53033	0.53033	0	0.15	0.15	0.15

Nummer	rel. Dichte	abs. Dichte	mx	my	mz	rx	ry	rz
11	0.3	1.3	0	0.75	0	0.15	0.15	0.15
12	0.4	1.4	-0.53033	0.53033	0	0.15	0.15	0.15
13	-0.1	0.9	-0.75	0	0	0.15	0.15	0.15
14	-0.2	0.8	-0.53033	-0.53033	0	0.15	0.15	0.15
15	-0.3	0.7	0	-0.75	0	0.15	0.15	0.15
16	-0.4	0.6	0.53033	-0.53033	0	0.15	0.15	0.15
17	1	2	0.39	0	0	0.01	0.01	0.01
18	1	2	0.378232	0.07187	0	0.015	0.015	0.015
19	1	2	0.349903	0.148216	0	0.02	0.02	0.02
20	1	2	0.301068	0.223569	0	0.025	0.025	0.025
21	1	2	0.229416	0.29029	0	0.03	0.03	0.03
22	1	2	0.135487	0.338922	0	0.035	0.035	0.035
23	1	2	0.023927	0.359204	0	0.04	0.04	0.04
24	1	2	-0.095582	0.34189	0	0.045	0.045	0.045
25	1	2	-0.20821	0.281333	0	0.05	0.05	0.05
26	1	2	-0.29525	0.178472	0	0.055	0.055	0.055
27	1	2	-0.337209	0.043476	0	0.06	0.06	0.06
28	1	2	-0.31874	-0.103101	0	0.065	0.065	0.065
29	1	2	-0.234702	-0.231981	0	0.07	0.07	0.07
30	1	2	-0.095661	-0.310603	0	0.075	0.075	0.075
31	1	2	0.069682	-0.312321	0	0.08	0.08	0.08
32	1	2	0.217855	-0.227518	0	0.085	0.085	0.085
33	1	2	0.301342	-0.072754	0	0.09	0.09	0.09
34	0.3	1.3	0.106066	0.106066	0	0.05	0.05	0.05
35	-0.054	0.946	-0.016795	0.149057	0	0.05	0.05	0.05
36	-0.424	0.576	-0.127009	0.079805	0	0.05	0.05	0.05
37	-0.79	0.21	-0.141582	-0.049542	0	0.05	0.05	0.05
38	3.2	4.2	-0.049542	-0.141582	0	0.05	0.05	0.05
39	-0.07	0.93	0.079805	-0.127009	0	0.05	0.05	0.05
40	-0.12	0.88	0.149057	-0.016795	0	0.05	0.05	0.05

Literaturverzeichnis

[1] C. Axelsson, *Direct Fourier Methods in 3D-Reconstruction from Cone-Beam Data*, Dissertation, Linköping University, Sweden, 1994

[2] C. Axelsson, P.E. Danielsson, *Three-dimensional reconstruction from cone-beam data in O($N^3 \log N$) time*, Phys. Med. Biol., Vol. 39 (1994), S. 477-491

[3] R. Clack, *Towards a complete description of three-dimensional filtered backprojection*, Phys. Med. Biol., Vol. 37 (1992), S. 645-660

[4] M. Defrise, R. Clack, *A Cone-Beam Reconstruction Algorithm Using Shift-Variant Filtering and Cone-Beam Backprojection*, IEEE Trans. Med. Imag., Vol. 13 (1994), S. 186-195

[5] M. Defrise, P.E. Kinahan, D.W. Townsend, C. Michel, M. Sibomana, D.F. Newport, *Exact and Approximate Rebinning Algorithms for 3-D PET Data*, IEEE Trans. Med. Imag., Vol. 16 (1997), S. 145-158

[6] R.L. Dietz, *Die Approximative Inverse als Rekonstruktionsmethode in der Röntgen-Computertomographie*, Dissertation, Universität des Saarlandes, 1999

[7] L.A. Feldkamp, L.C. Davis, J.W. Kress, *Practical cone-beam algorithm*, J. Optical Society of America, Vol. 1, No. 6, 1984, S. 612-619

[8] D.V. Finch, *Cone Beam Reconstruction with Sources on a Curve*, Department of Mathematics, Tufts University, MA, Oregon State University, OR

[9] M. Frigo, S.G. Johnson et al, *The FFTW-Library*, Software and Documentation via "http://www.fftw.org", MIT

[10] E. Gamma, *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*, Bonn: Addison-Wesley, 1996

[11] E.A. Hoxter, A. Schenz, *Röntgenaufnahmetechnik*, Berlin: Siemens AG, 1991

[12] H.M. Hudson, R.S. Larkin, *Accelerated Image Reconstruction Using Ordered Subsets of Projection Data*, IEEE Trans. Med. Imag., Vol 13 (1994), S. 601-609

[13] ISCA, CDI, interner Bericht 2001

[14] A.C. Kak, M. Slaney, *Principles of Computerized Tomographic Imaging*, New York: IEEE Press, 1988

[15] W.A. Kalender, *Computertomographie*, München: Publicis MCD Verlag, 2000

[16] H. Kudo, F. Noo, M. Defrise, *Quasi-Exact Filtered Backprojection Algorithm for Long-Object Problem in Helical Cone-Beam Tomography*, IEEE Trans. Med. Imag., 19(2000), S. 902-921

[17] H. Kudo, T. Saito, *Derivation and Implementation of a Cone-Beam Reconstruction Algorithm for Nonplanar Orbits*, IEEE Trans. Med. Imag., Vol. 13 (1994), S. 196-211

[18] J.V. Leahy, K.T. Smith, D.C. Solmon, *Uniqueness, Nonuniqueness, and Inversion in the X-ray and Radon Problems*, University of Oregon, 1979

[19] R.M. Lewitt, G. Muehllehner, J.S. Karp, *Three-dimensional image reconstruction for PET by multi-slice rebinning and axial image filtering*, Phys. Med. Biol., Vol. 39 (1994), S. 321-339

[20] A.K. Louis, *Analytische Methoden in der Computer-Tomographie*, Habilitationsschrift, Fachbereich Mathematik der Universität Münster, 1981

[21] A.K. Louis, *Inverse und schlecht gestellte Probleme*, Stuttgart: Teubner, 1989

[22] A.K. Louis, P. Maass, *Contour Reconstruction in 3-D X-Ray CT*, IEEE Trans. Med. Imag., Vol. 12(1993), S. 764-769

[23] A.K. Louis, P. Maass ,A. Rieder, *Wavelets: Theorie und Anwendungen*, Stuttgart: Teubner, 1994

[24] A.K. Louis, *Approximate inverse for linear and some nonlinear problems*, Inverse Problems, 12 (1996), S. 175-190

[25] A.K. Louis, T. Schuster, *A novel filter design technique in 2D computerized tomography*, Inverse Problems, Vol. 12 (1996), S,. 685-696

[26] A.K. Louis, *A Unified Approach to Regularization Methods for Linar Ill-Posed Problems*, Inverse Problems, Vol. 15 (1999), S. 489-498

[27] P. Maass, *The x-ray transform: singular value decomposition and resolution*, Inverse Problems, Vol. 3 (1987), S. 729-741

[28] P. Maass, *Die Singulärwertzerlegung der Röntgentransformation und ihre Anwendung in der Computertomographie*, Dissertation, Technische Universität Berlin, 1988

[29] M. Magnusson, *Linogram and Other Direct Fourier Methods for Tomographic Reconstruction*, Dissertation, Linköping University, Sweden, 1993

[30] H. Matsuo, A. Iwata, I. Horiba, N. Suzumura, *Three-Dimensional Image Reconstruction by Digital Tomo-Synthesis Using Inverse Filtering*, IEEE Trans. Med. Imag., Vol 12(1993), S. 307-313

[31] F. Natterer, *The mathematics of computerized tomography*, Stuttgart: Teubner, 1986

[32] W.H. Press et al, *Numerical recipes in C*, Cambridge: Cambridge Univerity Press, 1992

[33] A. Rieder, T. Schuster, *The Approximate Inverse in Action with an Application to Computerized Tomography*, SIAM J. Numer. Anal., Vol. 37 (2000), S. 1909-1929

[34] A. Rieder, *Principles of Reconstruction Filter Design in 2D-Computerized Tomography*, Proceedings "Joint Summer Research Conference on Radon transforms and tomography", 2000

[35] A. Rieder, R. Dietz, T. Schuster, *Approximate Inverse meets Local Tomography*, Math. Meth. Appl. Sci, 23 (2000), S. 1373-1397

[36] J. Rumbaugh, M. Blaha, W. Premerlani, F.Eddy, W.Lorensen, *Object-oriented Modeling and Design*, Englewood Cliffs: Prentice Hall, 1991

[37] N. Schmeisser, *Einführung in das Objektorientierte Programmieren*, Seminarunterlagen, Forschungszentrum Rossendorf e.V., Abt. Kommunikation und Datenverarbeitung, 1997

[38] U. Schmitt, *Effiziente Verfahren zur Regularisierung dynamischer inverser Probleme*, Dissertation, Universität des Saarlandes, 2001

[39] W. Schroeder, K. Martin, W. Lorensen, *The visualization toolkit*, New Jersey: Prentice Hall, 1997

[40] M.M. Seger, *Three-dimensional reconstruction from cone-beam data using an efficient Fourier technique combined with a special interpolation filter*, Phys. Med. Biol., Vol. 43 (1998), S. 951-959

[41] B. Stroustrup, *Die C++-Programmiersprache*, Bonn, München, Paris: Addison-Wesley, 1992

[42] P. Toft, *The Radon Transform*, Ph.D. Thesis, Technical University of

[43] H.K. Tuy, *An Inversion Formula for Cone-Beam Reconstruction*, SIAM J. Appl. Math., Vol 43 (1983), S. 546-552

[44] T. Veldhuizen et al, *Blitz++: Object-Oriented Scientif Computing*, Software and Documentation via "http://www.oonumerics.org/blitz", Denmark, May 1996

[45] X. Yan, R.M. Leahy, *Cone beam tomography with circular, elliptical and spiral orbits*, Phys. Med. Biol., Vol. 37 (1992), S. 493-506

[46] G.L. Zeng, R. Clack, G.T. Gullberg, *Implementation of Tuy's cone-beam inversion formula*, Phys. Med. Biol., Vol. 39 (1994), S. 493-507