Aus dem Institut für Medizinische Physik
der Medizinischen Fakultät
an der Friedrich-Alexander-Universität Erlangen-Nürnberg
Direktor: Prof. Dr. W. A. Kalender

Hochperformante Spiral-CT Bildrekonstruktion auf Many-Core Prozessoren mit Vektorrecheneinheit

Inaugural-Dissertation
zur Erlangung der Doktorwürde
der Medizinischen Fakultät
der
Friedrich-Alexander-Universität
Erlangen-Nürnberg

vorgelegt von
Sven Steckmann
aus
Schwabach

Gedruckt mit Erlaubnis
der Medizinischen Fakultät der
Friedrich-Alexander-Universität
Erlangen-Nürnberg

Dekan:	Prof. Dr. med. Dr. h.c. Jürgen Schüttler Klinik für Anästhesiologie
Referent:	Prof. Dr. Marc Kachelrieß Institut für Medizinische Physik
1. Koreferent:	Prof. Dr. med. Stephan Achenbach Medizinische Klinik 2 (Kardiologie)
2. Koreferent:	Prof. Dr. Peter Steffen Lehrstuhl für Multimediakommunikation und Signalverarbeitung

Tag der mündlichen Prüfung: 16. Dezember 2009

Ich widme diese Arbeit meinem verstorbenem Vater,
der in mir die naturwissenschaftliche Neugier weckte.

Inhaltsverzeichnis

1. Zusammenfassung

Hintergrund und Ziele: Die Computertomographie (CT) zählt trotz ihrer Strahlenexposition zu einem der wichtigsten bildgebenden Verfahren in der Medizin, da sie schnell und preiswert zuverlässige Bilder für die Diagnose liefert. Mit der Einführung der Spiral–CT vor 20 Jahren haben sich neue Möglichkeiten für den klinischen Einsatz ergeben. Heute werden mit ihr schätzungsweise 95% der CT–Aufnahmen für die klinische Routine erstellt. Daher ist die Spiral–CT immer noch ein interessantes Gebiet für Forschungstätigkeiten. Den Sprung aus der Wissenschaft in die Anwendung kann ein Algorithmus jedoch nur schaffen, wenn dieser in der klinischen Routine einsetzbar ist. Dort spielen dann Kriterien wie preiswerte Hardware und angemessene Rekonstruktionszeit eine entscheidende Rolle. Diese Kriterien sind oftmals nicht einfach zu erfüllen, da viele aus dem akademischen Feld stammende Algorithmen auf iterative Ansätze setzen und daher bereits ein Vielfaches an Rechenleistung im Vergleich zu heute eingesetzten einfachen gefilterten Rückprojektionsalgorithmen benötigen. Zur gleichen Zeit stellt die Industrie Computerhardware bereit, die immer leistungsfähiger wird. Hierfür wird zunehmend auf mehrere Rechenkerne und Vektorrechner gesetzt. Vektorrechner führen auf den Vektor, der aus mehreren Datenelementen besteht, den gleichen Befehl aus. Diese Rechner liefern mit dem klassischen Vorgehen bei der gefilterten Rückprojektion nur eine unzureichende Leistung, da eine Anpassung auf Vektorrechnung nur unzureichend möglich ist. In dieser Arbeit wird daher ein neuer, allgemein gültiger Ansatz für einen Algorithmus gesucht, wie sich eine Spiral–CT Rückprojektion und Vorwärtsprojektion auf Vektorrechenwerke anpassen lässt, um auch hier eine hohe Leistung zu erreichen.

Methoden: Um die Grundlagen für den Algorithmus zu schaffen, ist zunächst eine theoretische Betrachtung der Problemstellung „Spiral–CT Rückprojektion“ notwendig. Dabei wird herausgearbeitet, wie diese Algorithmen auf der Hardware ausgeführt werden und welche Probleme dabei auftreten. Dies wird anhand von momentan gut verfügbarer Standardhardware untersucht. Exemplarisch wird als Vertreter der PC–Architektur die neue Nehalem–Architektur von Intel, die in aktuellen PCs und Servern zum Einsatz kommt, verwendet. Als Architektur, die den Many–Core Ansatz sehr stark vertritt, wird die Cell–Architektur von IBM, Toshiba und Sony verwendet. Bei Many–Core Systemen wird die zur Verfügung stehende Leistung durch viele, unabhängige Rechenkerne

erreicht. Neben Servern findet der Cell auch einen breiten Einsatz in der Sony Playstation 3. Der hier vorgestellte generische Algorithmus beschreibt, wie die Symmetrien in der Spirale genutzt werden können, um den Anforderungen moderner Vektorrechner gerecht zu werden. Da hier ein generischer Algorithmus für eine Spiral-Rückprojektion vorgestellt wird, kann dieser durch viele Algorithmen eingesetzt werden, ohne dass diese Algorithmen angepasst werden müssen. Beispiele für Algorithmen, die durch den neuen Ansatz profitieren können, finden sich bei den exakten Bildrekonstruktionslagorithmen [23] ebenso wie iterative Algorithmen [1, 2] oder auch die auf der gefilterten Rückprojektion basierenden Algorithmen [17, 42]. Um die dabei erreichbare hohe Leistung zu demonstrieren wird exemplarisch eine Implementierung vorgenommen und am Beispiel des Algorithmus EPBP (Extended Parallel Backprojection) [17] für den Einsatz in der CT evaluiert.

Ergebnisse und Beobachtungen: Um die neue Methode zu evaluieren wurde die Bildqualität und die Leistungsfähigkeit auf den zur Verfügung stehenden Rechnern betrachtet. Dabei zeigte sich, dass der neue Ansatz keine negativen Auswirkungen auf die Bildqualität der Rekonstruktion hat. Mit der Standard-Rückprojektion benötigt aktuell eine Volumenrückprojektion von $512 \times 512 \times 512$ auf der Intel Nehalem Architektur 31,3 s, mit der neuen Methode reduziert sich diese Zeit auf 2,84 s. Dies entspricht einer Beschleunigung um den Faktor 11 und zeigt die hohe Leistung, die durch den neuen Ansatz erreicht werden kann. Die Beschleunigung kann ausgehend von diesem konservativen Wert noch deutlich höher ausfallen. Einen hohen Einfluss hat hier auch die Geometrie des Scanners. Der Ansatz zeigt sowohl auf der PC-Architektur, wie auch auf der Cell–Architektur eine sehr hohe Leistung. Durch die Nutzung der Spiral-Symmetrie wird zum einen eine vollständige Vektorisierung erreicht und zum Anderen werden Berechnungen eingespart.

Schlussfolgerungen: Erstmals ist es gelungen, eine Rückprojektion für die Spiral–CT vollständig und effektiv zu vektorisieren. Daraus ergibt sich auf einer Hardware, die von vektorieller Verarbeitung profitiert, eine hohe Leistung. Als großer Vorteil ist zu sehen, dass der neue Algorithmus generisch ist, wovon nahezu alle Rekonstruktionsalgorithmen profitieren, die auf eine Spiral-Rückprojektion angewiesen sind. In der gleichen Art und Weise lässt sich ein äquivalenter Vorwärtsprojektor erstellen, der in Zusammenarbeit mit dem Rückprojektor für iterative Rekonstruktionen benötigt wird. Damit wird für momentan noch akademisch eingesetzte, oftmals iterative Algorithmen, eine Chance eröffnet, den Weg in den klinischen Alltag zu finden. Diese Algorithmen versprechen dabei eine bessere Bildqualität, wurden jedoch bis jetzt aufgrund der Rechenzeit nicht eingesetzt.

2. Einleitung

Bereits aus dem Namen „Computertomographie“ (CT) ist ersichtlich, dass Computer und Rechenwerke einen starken Beitrag zum Funktionieren dieser Technologie leisten. Der erste Computertomograph, der vom Elektrotechniker Godfrey Newbold Hounsfield auf Basis der wenige Jahre vorher veröffentlichten Theorie von Allan McLeod Cormack um das Jahr 1969 entwickelt wurde, benötigte neben 9 Tagen für die Datenaufnahme noch 2 Stunden für die Berechnung der Schichtbilder. Heutzutage können durch das rasante Wachstum der Rechenleistung und der Entwicklung neuer Algorithmen moderne CT Geräte mit einer Verarbeitungszeit im Minutenbereich aufwarten, obwohl die Datenmengen stark gestiegen sind. Nicht zuletzt diese Eigenschaft hat für den Durchbruch der Computertomographie zu einem der wichtigsten bildgebenden Verfahren im klinischen Alltag geführt.

Die Entwicklung in der medizinischen Spiral-CT beschäftigt sich heutzutage unter anderem mit der Abdeckung eines größeren Volumens in kürzerer Zeit. Um dies zu erreichen, gibt es mehrere Möglichkeiten. Eine hiervon ist eine schnellere Rotation der aufnehmenden Gantry bei den Spiral-CT Geräten der 3. Generation. Hierbei werden jedoch mechanische Grenzen erreicht, die keine höhere Rotationsgeschwindigkeit mehr erlauben. Als Alternative wurden Geräte entwickelt, die ein größeres Volumen pro Umdrehung abdecken. Dies lässt sich durch die Vergrößerung des Detektors in Richtung der Patientenlängsachse erreichen. Dadurch ergeben sich jedoch neue Probleme bei der Rekonstruktion. Es lassen sich nicht mehr die bewährten und effizienten Algorithmen zur 2D-Rückprojektion, wie das ASSR [18] nutzen, denn diese lassen sich nur bei wenigen Detektorzeilen befriedigend einsetzten. Ansonsten müssen, wenn die Bildqualität erhalten bleiben soll, sogenannte 3D-Rekonstruktionen eingesetzt werden. Diese haben jedoch alle den Nachteil, dass sie um einen Faktor von 10 bis 50 rechenaufwändiger sind.

Abseits der Medizin erobert sich die Computertomographie neue Märkte in der Industrie wie auch in der Forschung. Zunehmend findet sie auch Einsatz bei der Gepäckkontrolle am Flughafen, in der Messtechnik und in der Archäologie, um nur einige Anwendungsgebiete zu nennen. Dabei kommen immer neue Herausforderungen auf die medizinischen, wie auch die nicht medizinischen Einsatzgebiete zu. Für all diese Einsatzgebiete ist es

essentiell, dass die Bilder in einer möglichst kurzen Zeit in einer angemessenen Qualität vorliegen. Dies stellt die Hersteller von Plattformen für die Rekonstruktion vor neue Herausforderungen. In der Archäologie ist eine etwas längere Wartezeit sicher noch zu vertreten, wohingegen bei der Gepäckkontrolle dies ein „No-go“ ist.

Leistungsfähige Rechnersysteme und Algorithmen sind also essentiell für die Weiterentwicklung. Es vergeht kaum ein Monat, in dem nicht eine neue Hardwareplattform angekündigt wird, die einen wesentlichen Leistungsvorsprung verspricht. Gordon Moore prägte den Begriff des Mooreschen Gesetzes. Heutzutage verstehen wir darunter, dass sich alle 18 Monate die Transistoranzahl pro Fläche verdoppelt. Dies entspricht für die Rechenleistung ebenso nahezu einer Verdopplung. Damit lässt sich annehmen, dass nach 5 Jahren die Rechenleistung um den Faktor 10 ansteigt. Dies lässt sich noch weiter steigern, indem nicht nur mehr Schaltkreise untergebracht werden, sondern auch die eigentlichen Rechenwerke effizienter gestaltet werden. Ein immer häufigerer Ansatz hierzu ist die Verwendung von Single Instruction Multiple Data (SIMD) Befehlen. Bei diesen Befehlen wird der gleiche Befehl auf mehrere Daten gleichzeitig ausgeführt. Sie werden oft auch Stream- oder Vektorbefehle genannt. Dieser Ansatz ist jedoch mit Nachteilen verbunden. Es wird erheblich komplexer die Recheneinheiten gut auszulasten, da die Flexibilität darunter leidet. Nachdem die darauf folgenden GHz-Schlachten der Hersteller ein jähes Ende gefunden hatten, wurden neue Wege der Effizienzsteigerung der Hersteller eingeschlagen. Mehrere Kerne kamen in Mode, denn die doppelte Anzahl an Kernen verspricht auch die doppelte Leistung. Für den Programmierer und das Design der Algorithmen bedeutet dies jedoch einen Mehraufwand, damit sich dies auch in der gemessenen Leistung niederschlägt.

Das erste Jahrzehnt des 21. Jahrhunderts ist geprägt durch zahlreiche neue alte Ideen. Neue Architekturen wurden erschaffen, so ist eine dieser Neuschöpfungen der Cell Prozessor, der bei seiner Vorstellung eine beeindruckende Leistung im Vergleich zu anderen Mikroprozessoren vorweisen konnte [16]. Die Nutzung der großen Leistung von GPUs (Graphical processing unit) wird nun effektiv durch eine Öffnung der Plattform von NVIDIA für Programmierer erreicht. Eine neue Architektur, der Larrabee, wird von Intel angekündigt und stark vermarktet. Dieser soll ebenso wie der Cell Prozessor auf kleine, spezialisierte und einfache Prozessorkerne zurückgreifen.

Dabei stellen sich oftmals die Fragen: Was können diese Plattformen leisten? Wie geeignet sind diese Plattformen für CT Rekonstruktionsalgorithmen? Bei der Entwicklung von Algorithmen für die CT wird jedoch oftmals nur die wissenschaftliche Seite betrachtet, die Implementierung ist zweitrangig. Dadurch haben viele aus dem wissenschaftlichen Bereich bekannten Algorithmen eine viel zu hohe Laufzeit für den praktischen Einsatz in der klinischen Routine.

Ziel und Aufbau der Arbeit

Die Industrie stellt Computerhardware bereit, die immer leistungsfähiger wird. Hierfür wird zunehmend auf mehrere Rechenkerne und Vektorrechner gesetzt. Vektorrechner führen auf den Vektor, der aus mehreren Datenelementen besteht, den gleichen Befehl aus. Diese Rechner liefern mit dem klassischen Vorgehen bei der gefilterten Rückprojektion nur eine unzureichende Leistung, da eine Anpassung auf Vektorrechnung nur unzureichend möglich ist. In dieser Arbeit wird daher ein neuer, allgemein gültiger Ansatz für einen Algorithmus gesucht, wie sich eine Spiral–CT Rückprojektion auf Vektorrechenwerke anpassen lässt, um auch hier eine hohe Leistung zu erreichen.

Die Arbeit gliedert sich in einen Teil mit den Grundlagen, der zunächst die CT-Rekonstruktion und besonders die dort verwendete Rückprojektion näher betrachtet. Im Anschluss daran werden die wichtigsten heute erhältlichen Hardwarearchitekturen und deren Eigenschaften am Beispiel des Cell Prozessors, der neuen Intel Nehalem Architektur und eines Grafikprozessors, des NVIDIA GTX 280 hinsichtlich ihrer Eigenschaften und deren Eignung für die Rekonstruktion diskutiert. Der Hauptteil besteht aus der Veröffentlichung mit der neuen, auf die Hardware zugeschnittenen Implementierung, die auf gedrehten Schichten basiert. Im Anschluss daran werden noch weiter gehende Ergebnisse bereit gestellt die den neuen Algorithmus und seine Eigenschaften näher betrachten.

3. Grundlagen

3.1. Computertomographie - vom Röntgenbild zu einem 3D Schichtbild

Die CT als bildgebendes Verfahren stellt ein Schichtbild zur Verfügung, das physikalisch auf dem Schwächungskoeffizienten $\mu(\boldsymbol{r}, E)$ an einem bestimmten Ort $\boldsymbol{r}(x, y, z)$ bei einer bestimmten Energie E basiert. In der Praxis werden jedoch nicht die Schwächungskoeffizienten angegeben, da diese vom Ort, wie von der Energie abhängig sind. Die später in den Bildern als Graustufen dargestellte CT-Zahl wird auf Wasser normiert, wodurch die Energie indirekt durch die Normierung festgelegt wird:

$$\text{CT-Zahl} = \frac{\mu - \mu_{\text{Wasser}}}{\mu_{\text{Wasser}}} \cdot 1000\,\text{HU} \tag{3.1}$$

Weiterführende CT-Grundlagen sind unter [21] zu finden.

3.1.1. Definition der Geometrie

Die ersten CT Scanner der 1. und 2. Generation hatten einen anderen Aufbau als die heutigen Scanner der 3. Generation. Dies spiegelt sich insbesondere in der Geometrie (siehe auch Abbildung 3.1) und dem verwendeten Messaufbau wieder. Die historischen Scanner verwendeten die Parallelstrahlgeometrie zur Aufnahme. Diese wird heute noch verwendet, allerdings nur auf der Rekonstruktionsebene, da sich Algorithmen leichter und zum Teil nur in dieser Geometrie formulieren lassen. Die Parallelstrahlgeometrie ist charakterisiert durch ein um den Winkel ϑ gedrehtes Koordinatensystem. Die Strahlen folgen immer entlang der η-Achse, die ξ-Achse dient nur zur Bestimmung eines Punktes im Raum. In der 3. Generation und der damit verbundenen Fächerstrahlgeometrie lässt sich ein Strahl durch den Drehwinkel der Röhre α und dem Winkel β für den Kanal eindeutig bestimmen. Eine Weiterentwicklung der Fächerstrahlgeometrie stellt die Kegelstrahlgeometrie dar. Dabei wird der Detektor in der z Richtung erweitert, diese Erweiterung wird als Zeile bzw. Schicht mit dem Formelzeichen b bezeichnet.

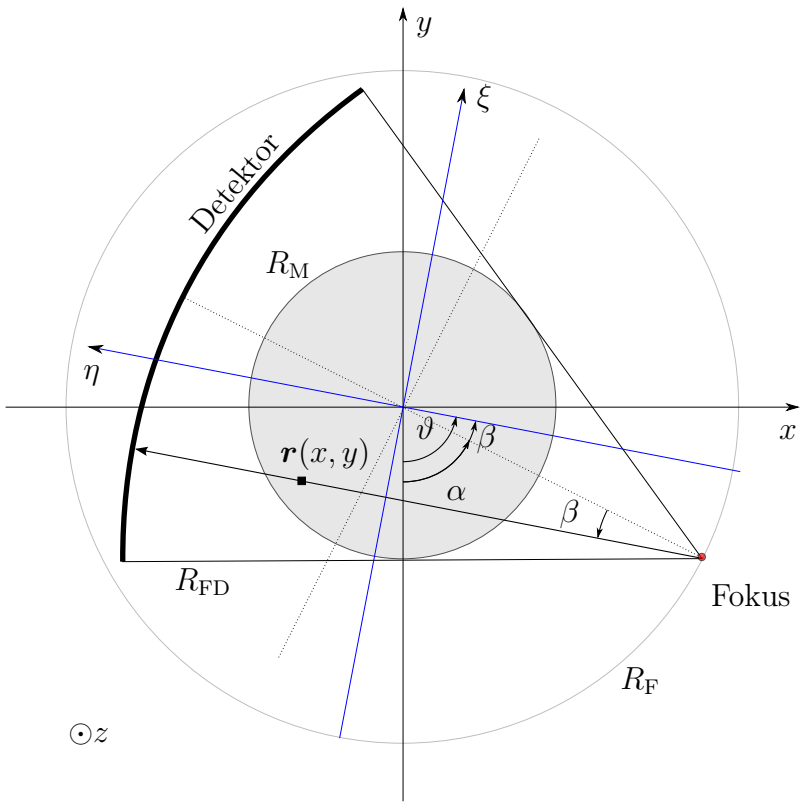


Abbildung 3.1.: Scanner der 3. Generation haben eine Röntgenröhre, die im Abstand R_{F} um das Zentrum rotiert. Der Detektor hat den Abstand R_{FD} von der Röhre, der gleichzeitig seinem Krümmungsradius entspricht. Zusammen kann diese Anordnung das Messfeld R_{M} komplett abdecken. Ein Strahl und seine Lage im Raum durch einen bestimmten Punkt $\boldsymbol{r}$ kann dabei ausreichend durch die Paare α und β in Fächerstrahlkoordinaten oder aber durch ϑ und ξ in Parallelstrahlkoordinaten beschrieben werden. Für die Ausdehnung des Detektors in z Richtung wird der Parameter b verwendet.

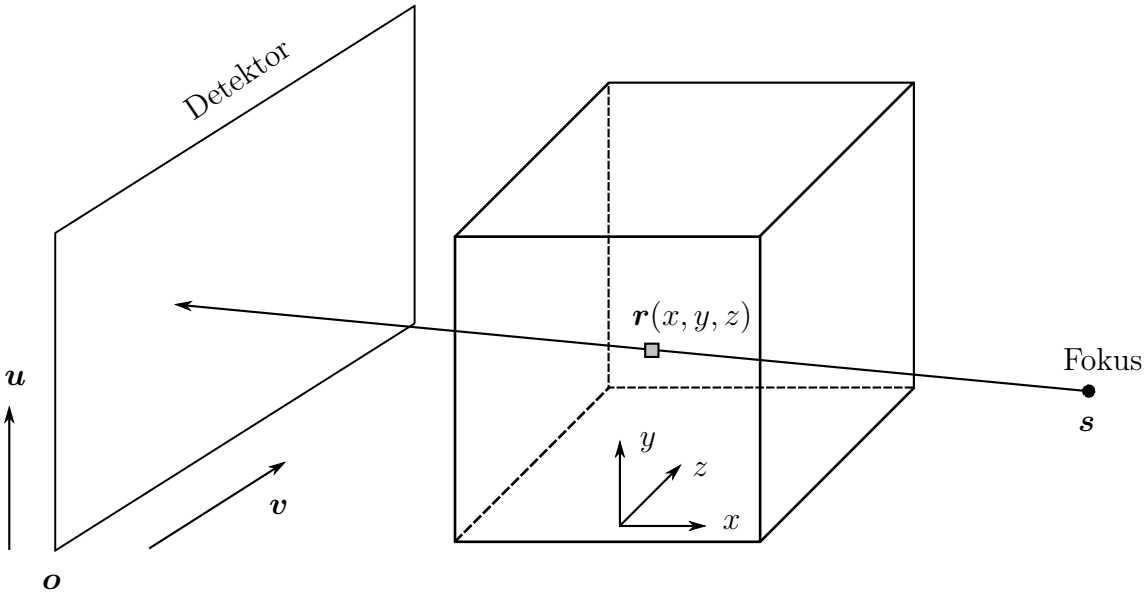


Abbildung 3.2.: Scanner in perspektivischer Geometrie mit einem Flachdetektor. Die Geometrie lässt sich vollständig durch die Neigung des Detektors mit $\boldsymbol{u}$ und $\boldsymbol{v}$, sowie einem Vektor $\boldsymbol{o}$ für den Detektorurspurng und einem Vektor $\boldsymbol{s}$ für den Fokus der Röntgenröhre beschreiben. Oftmals wird auch α für den Fokus verwendet, wobei dann gilt: $\boldsymbol{s}(\alpha)$. Der Vektor $\boldsymbol{r}$ beschreibt einen beliebigen Punkt im Raum des Volumens.

Eine der Spiral-Geometrie von der Rückprojektion sehr ähnliche Geometrie ist die perspektivische Geometrie, die ihre Anwendung in der Flachdetektor-CT findet (siehe Abbildung 3.2). Die perspektivische Geometrie verwendet im Gegensatz zu einem klinischen Scanner einen flachen Detektor. Ein Großteil von Anwendungen verwendet hierbei einen Kreis als Trajektorie für den Fokus der Röntgenröhre.

3.1.2. Röntgentransformation

Das Verfahren der Computertomographie beruht auf der physikalischen Grundlage der Schwächung von Röntgenstrahlung beim Durchgang durch Materie. Zu beachten ist, dass Röntgenstrahlung im Allgemeinen eine polyenergetische Strahlung ist. Die CT Rekonstruktion ist jedoch mit den gegenwärtigen bekannten Algorithmen nur möglich, wenn diese näherungsweise als monoenergetische Strahlung angenommen wird. Damit ergibt sich die Schwächung des Primärstrahles I_0 entlang der Linie L im Objekt $f(\boldsymbol{r})$ zu

$$I = I_0 \, \mathrm{e}^{-\int\limits_L \mu(\boldsymbol{r}) d\boldsymbol{r}} \,. \tag{3.2}$$

$\mu(\boldsymbol{r})$ ist dabei der lineare Schwächungskoeffizient und die Messgröße der CT. Im Folgenden wird dieser durch die Objektfunktion $f(\boldsymbol{r})$ ersetzt. Die Schwächung p und somit die Rohdaten sind die logarithmische Quotienten

$$p = -\ln \frac{I}{I_0} = \int_L f(\boldsymbol{r}) d\boldsymbol{r}. \tag{3.3}$$

Diese Transformation wird auch Röntgentransformation genannt und stellt die Grundlage für die Rekonstruktion der Daten dar. Für den 2D Fall der Datenaufnahme ist diese äquivalent zur Radontransformation [35]. Im 3D Fall ist dieser Zusammenhang nicht ohne weiteres herstellbar, so dass die Rekonstruktionsalgorithmen hier komplexer ausfallen oder nur approximativ arbeiten.

Ein Röntgenbild, wie es auch in der digitalen Radiografie aufgenommen wird, ist dabei nichts anderes, als die Messung von allen Schwächungswerten p, wobei die Linien L so gewählt sind, dass diese vom Fokus der Röntgenröhre auf den Röntgendetektor treffen. Ein Röntgenbild hat jedoch den Nachteil, dass eine Aussage über das $\mu(\boldsymbol{r})$ an einer bestimmten Stelle $\boldsymbol{r}$ im Körper nicht möglich ist. Um über $\mu(\boldsymbol{r})$ an einer bestimmten Stelle eine Aussage treffen zu können, rekonstruiert die CT aus den vorhandenen Schwächungsdaten p die Objektfunktion $f(\boldsymbol{r})$.

Für die Rekonstruktion genügen jedoch nicht einige wenige Projektionen. Jeder Voxel[1] im Bild muss aus einem Bereich von 0 bis $180°$ durchleuchtet werden. Allgemein wird dies in der Tuy-Smith Vollständigkeitsbedingung [43] beschrieben. Erreichen lässt sich dies durch eine Bewegung der Röntgenröhre und dem Detektor auf einer sinnvollen Trajektorie. Heutzutage haben sich hierfür hauptsächlich der Kreis und die Spirale etabliert.

3.1.3. Algorithmen für die Rekonstruktion

Nach der Datenaufnahme erfolgt die Rekonstruktion. Der Einsatz eines bestimmten Algorithmus ist dabei stark vom Anwendungsfall und von der Aufnahmesituation abhängig. Ferner haben die Verfahren selbst unterschiedliche Eigenschaften, was die Dosisnutzung und den Rechenaufwand angeht, sowie den Bildeindruck bei inkonsistenten oder verrauschten Daten. Mittlerweile ist das Feld der Methoden zur Rekonstruktion sehr umfangreich. Im Folgenden soll ein kurzer Überblick gegeben werden, der jedoch nicht den Anspruch auf Vollständigkeit hegt. Genaueres über die Grundlagen der Re-

[1] Ein Voxel ist ein 3 dimensionales, quadratisches Volumenelement.

konstruktion ist in [6, 19, 21] zu finden. Ein Vergleich verschiedener algebraischer und statistischer Algorithmen wurde in [3] durchgeführt.

Die Algorithmen lassen sich in mehrere Gruppen einteilen:

Approximative Algorithmen werden heute in den meisten Fällen eingesetzt. Sie schaffen nur ein ähnliches, approximatives Bild zu den Rohdaten. Wenn jedoch bestimmte Randbedingungen wie kleine Kegelwinkel erfüllt sind, gibt es nur geringe Abstriche in der Bildqualität. Typische Vertreter dieser Algorithmengattung sind Algorithmen wie ASSR [18] und AMPR [39]. Diese Algorithmen haben jedoch den Nachteil, dass sie bei größer werdendem Kegelwinkel immer schlechtere Bilder liefern. Dem kann man durch den Einsatz von 3D Algorithmen entgegenwirken. Diese haben jedoch einen um den Faktor 10 bis 50 höheren Rechenaufwand. Typische Vertreter für 3D Algorithmen sind für die Kreistrajektorie der FDK Algorithmus [9] sowie für die Spirale der EPBP Algorithmus [17] und die Methode mithilfe der Pi Fensterung [42].

Algebraische Algorithmen kamen vor allem in den Anfängen der CT zum Einsatz und erleben mittlerweile eine Renaissance. Sie basieren auf der Tatsache, dass sich die Projektionen als eine Linearkombination des zu rekonstruierenden Volumens darstellen. Das Finden dieser Linearkombinationen erfolgt über die Lösung eines linearen Gleichungssystems. Im Vergleich zu den approximativen Algorithmen kommen sie mit inkonsistenten Daten und ungleichmäßiger Geometrie besser zurecht. Typischer Vertreter ist ART [11,12] und seine Weiterentwicklung für eine effizientere Implementierung, SART [1, 19].

Statistische Algorithmen haben ihren Ursprung in der Nuklearmedizin. Dort sind es vor allem die EM- und ML-Methoden. Sie errechnen durch ähnliche Verfahren wie die Algebraischen Algorithmen eine Umkehrung der Radontransformation, allerdings unter Beachtung der Statistik der Aufnahmemodalität. In der CT hat sich hierfür der OSC Algorithmus [2] etabliert.

Exakte Algorithmen sind eine mathematisch eindeutige Möglichkeit, in der Kegelstrahl-CT Bilder zu rekonstruieren. Sie zeigen keine Artefakte wie sie bei den approximativen Algorithmen üblich sind. Sie haben hingegen den Nachteil, dass sie nur einen Teil der Dosis nutzen können, weswegen sie in der medizinischen CT noch ein Nischendasein fristen. Erste Grundlagen zur exakten Rekonstruktion wurden von Grangeat [13] gelegt. Die exakte Rekonstruktion ist ein noch immer offenes Forschungsfeld; zuletzt wurde durch die Inversionsformel nach Katsevich [23] ein großer Fortschritt verbucht. Eine Übersicht über die Entwicklungen gibt das Topical Review [44].

So unterschiedlich diese Verfahren auf den ersten Blick wirken, so haben sie fast alle eine Gemeinsamkeit: Sie benutzen eine Rückprojektion. Dies ist ein sehr rechenaufwändiger Schritt. Im Allgemeinen nimmt die Rückprojektion ca. 90% der Rekonstruktionszeit in Anspruch, ist jedoch ein sehr lokales und einfach zu beschreibendes Problem. Daher ist dies ein geeigneter Punkt, um die Optimierung auf eine bestimmte Hardware zu starten.

3.1.4. Rückprojektion - am Beispiel der gefilterten Rückprojektion

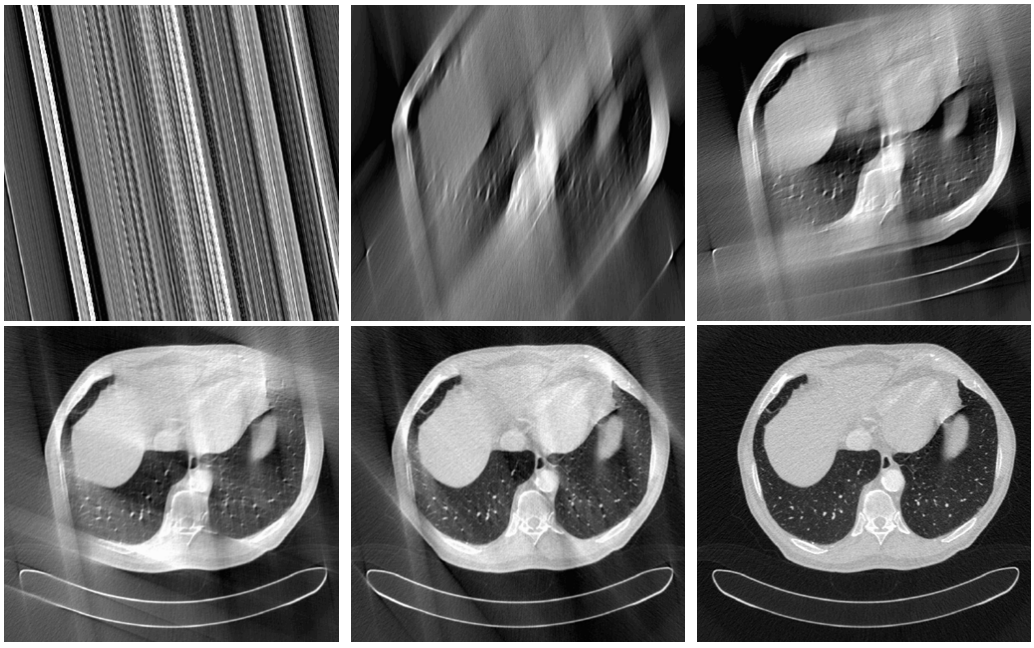


Abbildung 3.3.: Beispiel einer gefilterten Rückprojektion. In den Bildern variieren die bereits zurück projizierten Daten von 0° links oben bis 180° rechts unten. Dies entspricht dem fertigen Bild. Beim Bild mit 0° wurde nur eine Projektion verwendet, die Strahlen sind deutlich zu erkennen.

Eine Möglichkeit die Radontransformation umzukehren ist die gefilterte Rückprojektion. Die Herleitung ist in [6, 19] zu finden. Für die Parallelgeometrie ergibt sich diese zu

$$f(\boldsymbol{r}) = \int_0^{\pi} \int_{-\infty}^{\infty} P(\vartheta, u)\,|u|\,e^{2\pi j u \xi} du d\vartheta; \tag{3.4}$$

$$= \int_0^{\pi} p(\vartheta, \xi) * k(\xi) d\vartheta \tag{3.5}$$

Tabelle 3.1.: Korrespondenz der kontinuierlichen zu den diskreten Bezeichnern

Kontinuierlich	Diskret	Beschreibung
α	`n`	$0 \leq$ `n` $<$ `N` Anzahl der Projektionen
β	`m`	$0 \leq$ `m` $<$ `M` Detektorspalten in Fächerrichtung
b	`l`	$0 \leq$ `l` $<$ `L` Detektorzeilen in Kegelrichtung
$\boldsymbol{r}$, x	`i`	$0 \leq$ `i` $<$ `I` Anzahl der Voxel in x-Richtung
$\boldsymbol{r}$, y	`j`	$0 \leq$ `j` $<$ `J` Anzahl der Voxel in y-Richtung
$\boldsymbol{r}$, z	`k`	$0 \leq$ `k` $<$ `K` Anzahl der Voxel in z-Richtung
f	`Vol(i,j,k)`	Objektfunktion, Volumen in der Rekonstruktion
p	`Raw`	Rohdaten

mit

$$\xi(\boldsymbol{r}, \vartheta) \quad = \quad \boldsymbol{r} \cdot \begin{pmatrix} \cos\vartheta \\ \sin\vartheta \end{pmatrix} \tag{3.6}$$

Die Rohdaten $p(\vartheta, \xi)$ werden vor der Rückprojektion durch die Faltung mit dem Filterkern $k(\xi)$ gefiltert. Für diesen Kern gibt es verschiedene Implementierungen, die unterschiedliche Auswirkungen auf den Bildeindruck haben. Vom Grundtyp ist der Kern ein Hochpassfilter. Die bekannten Filter unterscheiden sich oftmals nur in der verwendeten Apodisierung. Bekannte Kerne sind der Ramlak [36] $K(u) = |u|$ und der Shepp-Logan [38] Kern.

Bei der Rückprojektion wird der gefilterte Wert des Strahls auf dessen Weg durch das Volumen zurückverfolgt und auf die dabei getroffenen Voxel aufaddiert. Wenn dies für alle Strahlen erfolgt ist, ergibt sich dann das fertige Volumen. Deutlich macht dies das Beispiel in der Parallelgeometrie aus Abbildung 3.3.

Für die Umsetzung auf einen Rechner muss die kontinuierliche Form aus Formel 3.5 zunächst in eine diskrete Form gebracht werden. Hierzu müssen die kontinuierlichen Bezeichner in diskrete umgewandelt werden. Dies wird mittels der Tabelle 3.1 durchgeführt. Im Anschluss ergibt sich dann

$$f(i,j) = \sum_{n=0}^{N-1} p_{\mathsf{c}}(n, m(i,j,n)) \tag{3.7}$$

als zu implementierender Algorithmus. Bei der Umsetzung für den Rechner ist es algorithmisch von Vorteil, wenn nicht nach den Strahlen vorgegangen wird, sondern nach den Voxeln. Diese Art von Rückprojektion wird auch voxelbasierte Rückprojektion genannt. Ein Beispiel einer Referenzimplementierung ist in Listing 1 zu finden. Hierbei wird für jeden Voxel die Position des Strahls auf dem Detektor ermittelt und der dor-

tige Wert nach einer Interpolation zwischen den benachbarten Detektorelementen auf das Volumen addiert.

Listing 1: Referenzimplementierung für eine Standard 2D-Rückprojektion in Parallelgeometrie.

```
void ParBackProjRef(
    int N, int M,                    // Anzahl der Projektionen, Kanäle
    int I, int J,                    // Anzahl der Voxel in x- und y-Richtung
    float **Slice, float **Raw,      // Arrays für Bilder und Rohdaten
    float *a, float *b, float *c )   // Koeffizienten für die Parallelgeometrie
{
    for( int n = 0; n < N; n++ )     // Schleife über die Projektionen
    for( int j = 0; j < J; j++ ) // Schleife über die Pixel in y-Richtung
    for( int i = 0; i < I; i++ ) // Schleife über die Pixel in x-Richtung
    {
        // Berechne Strahl durch den Voxel und den Auftreffpunkt auf dem Detektor
        float xi = a[n] * nx + b[n] * ny + c[n];
        int m = (int) floor( xi );
        // Berechne Gewicht für die lineare Interpolation
        float w = xi - m;
        // Addiere Detektorwert auf den Voxel
        if( m >=  0 && m <= M - 1 ) Slice[j][i] += ( 1 - w ) * Raw[n][m    ];
        if( m >= -1 && m <= M - 2 ) Slice[j][i] += w         * Raw[n][m + 1];
    }
}
```

In diesem Beispiel ist die Berechnung noch stark mit der Ausführung des Algorithmus verbunden. Werden beispielsweise nicht nur eine Schicht, sondern ein Schichtenstapel aus `K` Elementen rekonstruiert, dann müssen diese Berechnungen auch `K` mal durchgeführt werden. Daher ist es besonders bei oft wiederkehrenden Berechnungen von Vorteil, wenn die Ergebnisse bereits vorab in sogenannten Lookup-Tabellen gespeichert werden. Daneben verlagert dieses Vorgehen auch die Geometrieberechnungen, wodurch die Rückprojektion von der zugrunde liegenden Geometrie nahezu unabhängig wird.

3D Rückprojektion mit Lookup-Tabellen

Im vorherigen Abschnitt wurde die Rückprojektion von 2D Daten vorgestellt. Moderne Scanner arbeiten jedoch in der Kegelstrahlgeometrie und benötigen durch die dort verwendeten Algorithmen eine 3D Rekonstruktion. Die beiden Verfahren unterscheiden sich nur in der neu hinzugekommenen Dimension, die hier mit `l` der kontinuierlichen z-Ausdehnung b des Detektor und `k` für die zusätzliche Dimension von $\boldsymbol{r}$ repräsentiert wird. Zusätzlich wird noch ein Gewicht $w(n, i, j, k)$ notwendig. Dies wird im Folgenden Kapitel 3.1.5 ausführlicher erläutert.

Als Funktionen von $\boldsymbol{r}(\mathtt{i}, \mathtt{j}, \mathtt{k})$ und `n` stellen sich neben dem Gewicht `w` auch `m` und `l` dar. Sie werden durch die Geometrie des Scanners bestimmt und können entweder in

Lookup-Tabellen gespeichert oder während der Ausführung berechnet werden. Daraus ergibt sich dann die zu implementierende Summe:

$$f(i,j,k) = \sum_{n=0}^{N-1} p_{\mathsf{c}}(n,m,l) w(n,i,j,k) \tag{3.8}$$

Listing 2: Referenzimplementierung für eine Standard 3D-Rückprojektion.

```
void SpiralBP(  int const I, // Anzahl der x-Pixel      x
                int const J, // Anzahl der y-Pixel      y
                int const K, // Anzahl der Schichten    z
                int const L, // Anzahl der Zeilen       v
                int const M, // Anzahl der Kanäle       u
                int const N, // Anzahl der Projektionen a
                float const * const Raw,  // p(n, m, l)
                float       * const Vol ) // f(x, y, z)
{
    #define V(i, j, k)     Vol[( i * J + j ) * K + k]
    #define R(l, m, n)     Raw[L * M * n + L * m + l]

    for( int n = 0; n < N; n++ ) // theta-Schleife
    for( int i = 0; i < I; i++ ) // x-Schleife
    for( int j = 0; j < J; j++ ) // y-Schleife
    for( int k = 0; k < K; k++ ) // z-Schleife
    {
        int const   l = lCalc(n, i, j, k);   // v(a, x, y, z)
        int const   m = mCalc(n, i, j, k);   // u(a, x, y, z)
        float const w = wCalc(n, i, j, k);   // w(a, x, y, z)

        V(i, j, k) += w * R(l, m, n); // update Voxel
    }
}
```

In Listing 2 ist eine Standard Rückprojektion als C++ Code formuliert. Im hier vorliegenden Algorithmus werden `l`, `m` und das Gewicht `w` während der Rekonstruktion berechnet. Die Implementierung kann sowohl für die Spiral-Geometrie, als auch für die perspektivische Geometrie verwendet werden. Es müssen lediglich die Funktionen für die Berechnung von `l`, `m` und `w` gemäß der Formel 3.22 angepasst werden. Dies zeigt die enge Verwandtschaft dieser beiden Geometrien, weswegen sie auch die gleichen Probleme teilen.

3.1.5. Spiral-CT

Ein Meilenstein in der Geschichte der CT war 1989 die Einführung der Spiral-CT [20, 22]. Die Spiral-CT eignet sich vor allem dazu, um in kurzer Zeit ein großes Volumen aufzunehmen. Die heutigen klinischen Scanner arbeiten fast ausschließlich nach diesem Prinzip. Eine Spirale im Bezugssystem des Patienten entsteht, wenn der Patient während der Drehung der Gantry hindurch bewegt wird. Die Gantry ist ein drehbar gelagerter

Aufbau im CT Scanner, der die Röhre, wie auch den gegenüberliegenden Detektor enthält.

Die Bewegung des Patienten wird relativ zur Rotation der Gantry gemessen und mit d, dem Tischvorschub pro Umdrehung angegeben. Dieser Tischvorschub wird bei einer Aufnahme als konstant angenommen. Die Richtung der Bewegung ist parallel zur z Achse des Koordinatensystems.

Damit lässt sich die Trajektorie der Quelle wie folgt beschreiben:

$$\boldsymbol{s}(\alpha) = R_{\mathsf{F}} \begin{pmatrix} \sin(\alpha) \\ -\cos(\alpha) \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ 0 \\ d/2\pi \end{pmatrix}. \tag{3.9}$$

Die Position des Detektors lässt sich durch die Verknüpfung der Quelle und des Strahls gewinnen:

$$\boldsymbol{r}(\alpha, \beta, b) = \boldsymbol{s}(\alpha) + R_{\mathsf{FD}} \begin{pmatrix} -\sin(\alpha+\beta) \\ \cos(\alpha+\beta) \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{3.10}$$

Die Spirale in der Kegelstrahlgeometrie lässt sich durch folgende Transformation in die Parallelgeometrie überführen:

$$\vartheta = \alpha + \beta \tag{3.11}$$

$$\xi = -R_{\mathsf{F}} \sin\beta \tag{3.12}$$

Die Trajektorie der Quelle lässt sich dann durch

$$\boldsymbol{s}(\vartheta, \xi) = \boldsymbol{s}(\vartheta + \arcsin\frac{\xi}{R_{\mathsf{F}}}) \tag{3.13}$$

beschreiben. Für die spätere Rekonstruktion ist es wichtig, dass zu einem bestimmten Strahl, der durch die Quelle $\boldsymbol{s}$ und den durchstrahlten Voxel $\boldsymbol{r}$ bestimmt ist, die abhängigen Parameter ermittelt werden können. Diese lassen sich mittels geometrischer Transformationen aus dem Vektor $\boldsymbol{r}$ für die Voxelposition und dessen Komponenten r_x, r_y und r_z sowie aus dem Quellvektor $\boldsymbol{s}$ ermitteln und lauten wie folgt:

$$b = \frac{R_{\mathsf{FD}}(r_z - s_z)}{\sqrt{(r_x - s_x)^2 + (r_y - s_y)^2}} \tag{3.14}$$

$$\xi = r_x \cos\vartheta + r_y \sin\vartheta \tag{3.15}$$

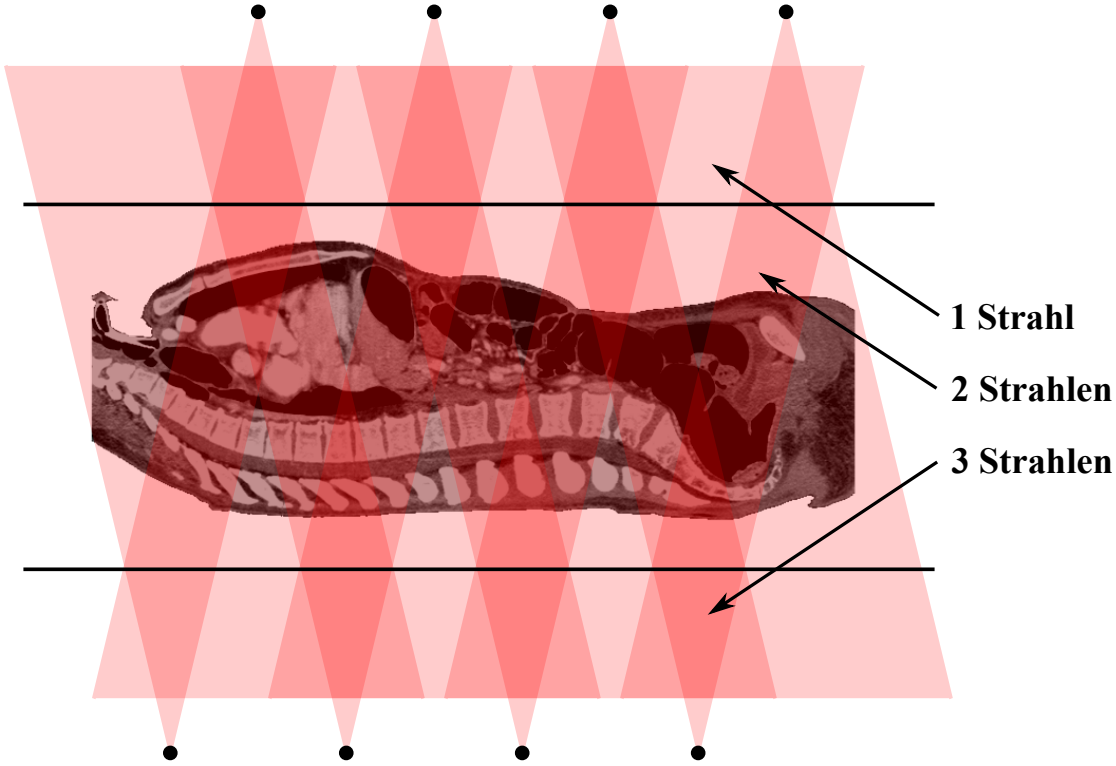


Abbildung 3.4.: Bei einer Spiral-CT Aufnahme werden Bereiche des Messobjekts unterschiedlich oft durchstrahlt. Hier ist ein Muster für den Winkel ϑ der Strahlen durch die mittlere Schicht dargestellt. Für eine möglichst hohe Dosisnutzung müssen diese Strahlen korrekt gewichtet werden. Eine Möglichkeit für dieses Gewicht ist jeweils der Kehrwert der Anzahl der Strahlen aus der jeweiligen Richtung ϑ durch den Voxel.

In der Spirale tritt im Vergleich zum Kreis noch ein zusätzliches Problem zu Tage, die Sichtbarkeit der Voxel und die Unterbrechung der Bestrahlung. Darunter wird verstanden, dass ein Voxel mehrmals durch den Röntgenstrahl getroffen werden kann. Dies ist auch in Abbildung 3.4 dargestellt. Dabei kommt es vor, insbesondere in den peripheren Voxeln, dass die Bestrahlung auch über einen Winkelbereich von α ausbleibt und anschließend wieder einsetzt, also unterbrochen ist. Dies macht es notwendig, insbesondere dann, wenn die volle Dosis genutzt werden soll, die Strahlen zu gewichten. Dies ist jedoch eine nicht triviale Aufgabe, die der Algorithmus EPBP [17] durch numerische Berechnung löst. Die Arbeit widmet sich jedoch nicht einem bestimmten Rekonstruktionsalgorithmus, vielmehr wird in dieser Arbeit eine allgemeine Methode für die Beschleunigung der Spiral-Rückprojektion vorgestellt. Um diese jedoch für den praktischen Einsatz zu evaluieren wird beispielhaft der Algorithmus EPBP implementiert und die so gewonnen Bilder mit denen der herkömmlichen Rückprojektions-Methode verglichen.

EPBP als Algorithmus für die Spirale

Der Algorithmus basiert auf der gefilterten Rückprojektion und verwendet die Parallelstrahlgeometrie. Diese wird durch vorherige Rebinningschritte hergestellt, da üblicherweise die Daten in der Fächerstrahlgeometrie aufgenommen werden. Für die Rekonstruktion werden daraufhin folgende Schritte durchgeführt:

Datenaufnahme und Längenkorrektur $p_0(\alpha, \beta, b) \cos\epsilon \longrightarrow p_1(\alpha, \beta, b)$

Azimuthales Rebinning $p_1(\alpha, \beta, b) \longrightarrow p_2(\vartheta, \beta, b)$

Radiales Rebinning $p_2(\vartheta, \beta, b) \longrightarrow p_3(\vartheta, \xi, b)$

Filterung durch Faltung $p_3(\vartheta, \xi, b) \longrightarrow p_c(\vartheta, \xi, b)$

Rückprojektion und Gewichtung $p_c(\vartheta, \xi, b) \longrightarrow f(\boldsymbol{r})$

Für eine hohe Qualität der Bilder ist eine Faltungsrichtung zu wählen, die tangential zur Trajektorie erfolgt. Dies ist besonders wichtig bei der Rekonstruktion von einem Spiralscan. Bei einem Kreis- oder Sequenzscan ist die Richtung der Kanäle (β) in der Projektionen ohnehin exakt die Faltungsrichtung. In der ursprünglichen Implementierung wurde diesem Umstand durch das so genannte longitudinale Rebinning Rechnung getragen. Bei näheren Untersuchungen stellte sich jedoch heraus, dass dieses longitudinale Rebinning größtenteils implizit durch den vorherigen durchgeführten Schritt des azimuthalen Rebinnings erfolgt. Dies lässt sich damit begründen, dass nach diesem Rebinning die Positionen der Quellen bei der Darstellung als Array immer exakt senkrecht auf dem Detektor, über der „Mittellinie“ des Detektors bzw. des Arrays angeordnet

sind. Es ergibt sich dadurch als Faltungsrichtung zwar nicht exakt die Tangente, jedoch ein Verlauf, der der Spiraltrajektorie und somit der Tangente in sehr guter Näherung folgt.

Für die eigentliche Rückprojektion wird der folgende Ausdruck einer gefilterten Rückprojektion benutzt:

$$f(\boldsymbol{r}) = \int p_{\mathsf{c}}(\vartheta, \xi, b) w(\vartheta, \boldsymbol{r}) \; d\vartheta \tag{3.16}$$

Die abhängigen Parameter ξ und b können durch die Formeln 3.15 und 3.14 ermittelt werden.

Im Vergleich zur einfachen parallelen Rückprojektion aus Kapitel 3.1.4, kommt hier zusätzlich ein Gewicht $w(\vartheta, \boldsymbol{r})$ zum Einsatz. In der Spirale gibt es für eine Richtung ϑ oft nicht nur genau einen Strahl, sondern mehrere. Diese Strahlen sind redundant und müssen entweder aussortiert oder aber entsprechend gewichtet werden. In der medizinischen CT ist ein Aussortieren nicht möglich, da dies dem Grundsatz der möglichst hohen Dosisnutzung widerspricht. Das Gewicht ist in der Spirale im Gegensatz zu Kreis- oder Sequenzscans nicht trivial bestimmbar, weswegen es numerisch berechnet werden muss.

Für diese Berechnung bekommen die Projektionen $p_{\mathsf{c}}(\vartheta, \xi, b)$ selbst zunächst ein Gewicht $w_{\mathsf{view}}(\vartheta)$ für ihre Relevanz. Dieses Gewicht spielt insbesondere in der Cardio-CT eine entscheidende Rolle. Die Gewichte sagen dabei aus, welche Qualität (die abgeleitet ist für welche Herzphase die Rekonstruktion durchgeführt werden soll) die Strahlen in dieser Projektion haben. Das Gewicht $p_{\mathsf{c}}(\vartheta, \xi, b)$ ist vom Voxel im Volumen unabhängig, denn es handelt sich um das Gewicht, wie diese Projektion in der Rekonstruktion gewichtet werden soll. Daraus lässt sich das voxelabhängige Gewicht berechnen, bei dem auch die Sichtbarkeit der Voxel mit einbezogen wird. Im Falle einer Standardrekonstruktion gilt $w_{\mathsf{view}}(\vartheta) = 1$.

Das Gewicht $w(\vartheta, \boldsymbol{r})$ muss folgenden Voraussetzung genügen, damit eine sinnvolle Rekonstruktion zustande kommt:

- Die Gesamtsumme aller Strahlen aus einer Richtung ϑ muss 1 ergeben:

$$\sum_{k=-\infty}^{\infty} w(\vartheta + k\pi, \boldsymbol{r}) = 1 \qquad \forall \vartheta. \tag{3.17}$$

- Die Gesamtnormierung muss π betragen:

$$\int_{-\infty}^{\infty} w(\vartheta, \boldsymbol{r}) \; d\vartheta = \pi. \tag{3.18}$$

- Es muss noch die Sichtbarkeit des Voxels (wann wird dieser auf den Detektor projiziert), als auch das Projektionsgewicht $w_{\mathsf{view}}(\vartheta)$ berücksichtigt werden.
- Von Seiten der der Signalverarbeitung bzw. der Numerik zeigt sich, dass sprunghaft implementierte Gewichtsfuktionen, insbesondere an den Rändern des Detektors zu Artefakten im Bild führen. Daher müssen die Gewichte einen sanften Auslauf an den Rändern erhalten.

Eine Möglichkeit um den Vorgaben zu genügen und sinnvolle Gewichte zu errechnen ist:

$$w_{\mathsf{sum}}(\vartheta, \boldsymbol{r}) = \sum_{k=-\infty}^{\infty} \begin{cases} w_{\mathsf{view}}(\vartheta + k\pi) & \text{falls Strahl den Voxel trifft} \\ 0 & \text{sonst} \end{cases}; \quad (3.19)$$

$$w(\vartheta, \boldsymbol{r}) = \frac{w_{\mathsf{view}}(\vartheta)}{w_{\mathsf{sum}}(\vartheta, \boldsymbol{r})}. \quad (3.20)$$

Wie hierbei zu erkennen ist, ist die Berechnung der Summengewichte $w_{\mathsf{sum}}(\vartheta, \boldsymbol{r})$ nur für einen Bereich von $\vartheta \in [0, \pi[$ notwendig. Durch eine geschickte neue Anordnung der Projektionen lässt es sich erreichen, dass diese Berechnung der Gewichte während der Rückprojektion durchgeführt werden kann. Dabei werden immer die Projektionen aus einem Winkel ϑ gemeinsam bearbeitet:

$$f(\boldsymbol{r}) = \int_0^{\pi} d\vartheta \; \frac{1}{w_{\mathsf{sum}}(\vartheta, \boldsymbol{r})} \sum_{k=-\infty}^{\infty} p_{\mathsf{c}}(\vartheta + k\pi, \xi, b) w_{\mathsf{view}}(\vartheta + k\pi) \quad (3.21)$$

Dies ist eine effiziente Möglichkeit zur Implementierung der Gewichtung, insbesondere wenn auch wenig Speicherplatz zur Verfügung steht. Es können jedoch, wenn entsprechende Ressourcen vorhanden sind, auch alternative Wege eingeschlagen werden. So kann auch eine Speicherung der Gewichte mit ihren Symmetrien sinnvoll sein. Dies wird im späteren Ansatz der Rekonstruktion mit gedrehten Schichten, wie er im Kapitel 4 vorgestellt wird, verwendet.

3.1.6. Kreisbahn als Trajektorie

Die Kreisbahn als Trajektorie ist einfacher als die Spirale zu realisieren. Sie kommt besonders wegen ihrer Flexibilität in der interventionellen Medizin und in der Technik

zum Einsatz. Hierbei spielt auch die Mikro-CT eine Rolle, bei der kleinste Objekte mit sehr hoher Ortsauflösung vermessen werden können.

Zur Rekonstruktion der Rohdaten wird heute meist der FDK-Algorithmus [9] verwendet. Dieser basiert auf dem Grundsatz der gefilterten Rückprojektion. Die Rückprojektion, welche hier Verwendung findet, ist eine perspektivische Transformation vom Detektor auf die jeweiligen Schichten. Die Geometrie für die perspektivische Rückprojektion ist in Abbildung 3.2 zu finden. Die perspektivische Transformation eines Punktes $\boldsymbol{r}(x, y, z)$ auf den Detektor in einer bestimmten Position im Raum lässt sich wie folgt darstellen:

$$\begin{aligned} u(x,y,z) &= (c_{00}x + c_{01}y + c_{02}z + c_{03})w(x,y,z) \\ v(x,y,z) &= (c_{10}x + c_{11}y + c_{12}z + c_{13})w(x,y,z) \\ w(x,y,z) &= (c_{20}x + c_{21}y + c_{22}z + c_{23})^{-1} \end{aligned} \tag{3.22}$$

Dabei sind die Koeffizienten c_{00} bis c_{23} die perspektivischen Koeffizienten, die sich aus den Vektoren $\boldsymbol{o}$, $\boldsymbol{u}$, $\boldsymbol{v}$ und $\boldsymbol{s}$ (siehe Abbildung 3.2) errechnen lassen.

3.2. Betrachtung des Rechen- und Datentransferaufwandes für die Rückprojektion und erste Optimierungsschritte

Für die Optimierung ist es wichtig zu wissen, wie groß der Aufwand des Algorithmus ist. Anhand dessen kann zum Einen der Grad der Optimierung abgeschätzt werden und zum Anderen die Strategie der Optimierung gewählt werden. Diese Betrachtung ist am Besten ausgehend von Formel 3.8 zu führen, da hier die essentiellen Rechenoperationen pro Voxel abgeschätzt werden können. Für die Betrachtung des Datentransfers und des Rechenaufwandes ist es notwendig, dass zunächst Optimierungen durchgeführt werden, da anderenfalls falsche Schlüsse gezogen werden. Grundsätzlich lassen sich zwei Formen der Optimierung unterscheiden: Die Verminderung des Rechenaufwandes und die Anpassung an die Hardware-Architektur. Für diese Betrachtung spielt es zunächst keine Rolle, wenn der Algorithmus noch nicht an die Hardware angepasst wurde, da diese Anpassung nichts am eigentlichen Rechenaufwand des grundlegenden Algorithmus ändert.

Auf Basis der Formel 3.8 ist der Grundaufwand das Laden und Gewichten eines Rohdatenwertes sowie der abschließenden Addition auf das Volumen. Dieser Vorgang wird auch als ein Update bezeichnet. Zum Grundaufwand des Updates kommt noch der Aufwand für die Bestimmung der Position auf dem Detektor, bei der der Rohdatenwert gelesen wird. Wenn Lookup-Tabellen für die Bestimmung des Gewichts eingesetzt werden, beträgt dieser Aufwand zwei lesende Zugriffe. Bei der Berechnung ist der Aufwand stark von der Geometrie und der Implementierung abhängig. Die Werte in Tabelle 3.2 sind sowohl für die nicht optimierte wie auch für die optimierte Rückprojektion gegeben. Die Basis ist für alle Rückprojektionen gleich. Zusätzlich zu dieser Basis kommen noch die für die Geometrie spezifischen Aufwände. Aufwände für Schleifen oder ähnliches werden in der Berechnung vernachlässigt. Für die optimierten Fassungen muss noch zwischen der x/y und der z-Richtung gesondert unterschieden werden, denn es lassen sich bei einer solchen gesonderten Unterscheidung zusätzliche Optimierungen vornehmen. Dies ist bei den optimierten Algorithmen mit dem Zusatz gekennzeichnet. Für die Betrachtung des zu erwartenden Rechenaufwands genügt es, wenn später nur die z-Richtung betrachtet wird, denn dies wird bei der späteren Implementierung die innerste Schleife darstellen. Diese Schleife wird im Verhältnis zu den anderen (x und y) um eine Ordnung mehr durchlaufen und ist somit kritisch für eine schnelle Verarbeitung. Die x/y-Richtung wird nur der Vollständigkeit halber angegeben.

Um den Erfolg der Optimierung zu messen, gibt es in der Literatur viele Ansätze. Dabei werden oftmals einige bestimmte Szenarien verwendet, die dann mit der zeitlichen Laufzeit vermessen werden. Dies ist jedoch zwischen verschiedenen Geometrien nur schwer vergleichbar. Auch kann die Effizienz bei einer Änderung der Geometrie nur schwer abgeschätzt werden. Daher wurde das Maß der GUPS (Giga Updates Per Second) erstmals in [10] eingeführt. Damit lassen sich verschiedene Algorithmen leichter miteinander vergleichen und die Benchmarks bedingt ineinander umrechnen. Das Ergebnis der Messung ergibt sich aus den GU (Giga Updates) und der Zeit für die Rekonstruktion:

$$\begin{aligned} GU &= \frac{N_{\text{Updateschritte}}}{1024^3}, \\ GUPS &= \frac{GU}{t_{\text{Rekonstruktion}}}. \end{aligned} \tag{3.23}$$

Optimierung bei der Geometrieberechnung

Für den Prozessor ist es ein hoher Aufwand, die Geometrie für den Rückprojektionsschritt in jedem Schleifendurchlauf für jeden Voxel komplett neu zu berechnen. Die Berechnung für die Geometrie entspricht einer perspektivischen Verzerrung der Projektion auf die Schicht. Dabei gilt es stets, einen Weg zu finden, diese Berechnungen so

zu vereinfachen, dass diese in Schritten durchlaufen werden können und Berechnungen aus vorherigen Schritten wiederverwendet werden können.

Beispielsweise ist die b-Richtung im klinischen Scanner immer so ausgerichtet, dass sie der z-Richtung im Volumen folgt. Damit ist es möglich, die Berechnungen aufzuspalten:

$$lCalc(n,i,j,k) \longrightarrow lCalc(n,i,j) + \Delta l(\Delta k) \tag{3.24}$$

$$mCalc(n,i,j,k) \longrightarrow mCalc(n,i,j) \tag{3.25}$$

Durch diese Vereinfachung lässt sich der Algorithmus schon wesentlich effizienter implementieren, da für eine Volumenstange in z-Richtung die Berechnung des m-Wertes komplett entfällt und sich beim l-Anteil für jeden Voxel ein effizient zu implementierendes Inkrement ergibt. Für weitere Informationen zu diesem Thema sei auf [25] verwiesen.

Kurze Betrachtung der Geometrien und deren speziellen Geometrievereinfachungen

Lookup-Tabelle Bei der Lookup-Tabelle sind grundsätzlich für einen Rohdatenwert zwei Zugriffe (Detektorposition und Gewicht) notwendig. Diese lassen sich reduzieren, wenn die Symmetrie in der Spirale genutzt wird. Dies stellen die Tabellenzeilen „Lookup-Tabelle HP Spirale“ dar. Dann wird nur noch bei einer x/y Veränderung ein Zugriff notwendig. Details zu diesem Algorithmus sind in Kapitel 4 zu finden.

Spirale Der Detektorzugriff in der Spirale setzt sich aus der Berechnung von ξ und b zusammen. Diese lassen sich mit folgenden Formeln berechnen:

$$\xi = r_x \cos\vartheta + r_y \sin\vartheta \tag{3.26}$$

$$b = \frac{R_{\mathsf{FD}}(r_z - s_z)}{\sqrt{(r_x - s_x)^2 + (r_y - s_y)^2}}. \tag{3.27}$$

r_x steht dabei für die x-Komponente von $\boldsymbol{r}$. $\boldsymbol{s}$ ist die Position vom Fokus der Röntgenröhre im Raum und s_x die entsprechende x-Komponente. Für die anderen Richtungen gilt dies analog. Für die optimierte Implementierung ist insbesondere die Wurzel störend. Da jedoch die Richtungen von b und z identisch sind, ist es möglich, diese Richtung abzuspalten. Dadurch entfallen viele Berechnungen oder sie lassen sich durch inkrementelle Berechnungen lösen. Das gleiche gilt für die trigonometrischen Funktionen, da diese nur für jeden neuen Projektionswinkel berechnet werden.

Tabelle 3.2.: Rechenoperationen der Algorithmen im Vergleich.

Algorithmus	Zusatz	Lesen	Schreiben	Addition	Multiplikation	Division	Sonstiges
Basis		2	1	1	1		
Lookup-Tabelle Spirale		2					
Lookup-Tabelle HP Spirale (opt.)	z						
	x/y	2					
Spirale				5	5	1	sin, cos, sqrt
Spirale (opt.)	z			1	0	0	
	x/y			4	4	1	sqrt
perspektivische				9	11	1	
perspektivische (opt.)	z			1	0	0	
	x/y			7	9	1	

Perspektivische Geometrie Die perspektivische Transformation wurde in Formel 3.22 vorgestellt. Ihre Implementierung ist besonders dann effizient zu erledigen, wenn eine Richtung des Detektors mit der z-Achse zusammenfällt. Denn dann vereinfacht sich die perspektivische Transformationsformel zu

$$\begin{aligned} u(x,y,z) &= (c_{00}x + c_{01}y + \quad\; c_{03})w(x,y,z) \\ v(x,y,z) &= (c_{10}x + c_{11}y + c_{12}z + c_{13})w(x,y,z) \\ w(x,y,z) &= (c_{20}x + c_{21}y + \quad\; c_{23})^{-1} \end{aligned} \tag{3.28}$$

und es wird möglich, die u-Richtung für eine gleich bleibende x- und y-Position konstant zu halten und die v-Richtung durch eine inkrementelle Berechnung zu ersetzen. Dadurch lässt sich wieder eine Dimension besonders schnell implementieren. Weiterführende Literatur ist unter [25] zu finden.

Fazit

Aus den vorangegangenen Überlegungen und der Tabelle 3.2 wird ersichtlich, welchen Aufwand die Rückprojektion hat. Dabei ist auffällig, dass die perspektivische und die Spiral-Rückprojektion in den jeweils optimierten Fassungen einen annähernd gleichen Aufwand erfordern. Bei der Spirale wird jedoch noch eine Gewichtung notwendig, die in dieser Betrachtung noch vernachlässigt wurde.

Interessant für die Abschätzung der Optimierungsstrategie ist die Verteilung zwischen Rechenoperationen und Datentransfer. Dabei ist ersichtlich, dass für die beiden optimierten Rückprojektionsarten ein Lesezugriff und zwei Schreibzugriffe notwendig sind. Demgegenüber stehen zwei Additionen und eine Multiplikation. Da heutige Rechner eher auf eine hohe Rechenleistung und nicht auf eine hohe Datentransferleistung ausgelegt sind, werden die weiteren Betrachtungen für die Optimierung für die Architektur sich hauptsächlich um den Datentransfer drehen.

3.3. Aufbau moderner Computerhardware und ihre Eigenschaften

Entwickler wie Einkäufer haben oftmals die Qual der Wahl, welche Rechner sie für eine bestimmte Aufgabe einsetzen. Dies hängt zum einen von der Stückzahl, aber insbesondere von der benötigten Leistung ab. Der Markt lässt sich dabei grob in einen Sektor für Standardhardware und einen für Spezialhardware einteilen. Dabei ist es von entscheidender Bedeutung, welche Anforderungen an die Hardware gestellt werden und wie sich das Entwicklerteam zusammensetzt. Mit dem Einsatz von Hardware, die auf CT-Rekonstruktion spezialisiert wurde oder sich dafür besonders eignet, lässt sich deutlich bessere Leistung erzielen, als dies mit Standardhardware möglich ist. Dabei ist jedoch ein gesonderter Entwicklungsaufwand oder spezielles Wissen über die jeweilige Hardware notwendig. Früher war die Rekonstruktion oftmals auf speziell entwickelte Hardware angewiesen. Mittlerweile ist die verfügbare Rechenleistung jedoch soweit gestiegen, dass solche Speziallösungen nur noch selten benötigt werden. Dennoch eignet sich nicht jede Architektur gleichsam für die gestellte Aufgabe, obwohl die Kosten ähnlich sind. Im Folgenden werden zwei Standard-Architekturen vorgestellt. Weiterhin sollen noch GPUs (Graphical Processing Units) auf Grafikkarten untersucht werden, da diese besonders bei iterativen Rekonstruktionen vermehrt eingesetzt werden.

3.3.1. Multicore Prozessoren mit der x86 Architektur

Eingeführt wurde diese Architektur und der auch heute noch verwendete Befehlssatz mit dem 80386 im Jahr 1985 von Intel, Santa Clara, USA [27]. Sie verhalf damals dem Personal Computer, kurz gesprochen PC, zu seinem Durchbruch. Dieser ist auch heute noch die Grundlage für die modernen Arbeitscomputer im Büroalltag. Seit ihrer Einführung hat die Architektur viele Erweiterungen erfahren und wird von vielen Prozessorherstellern als Grundlage verwendet. Sie ist eine relativ einfach zu programmierende

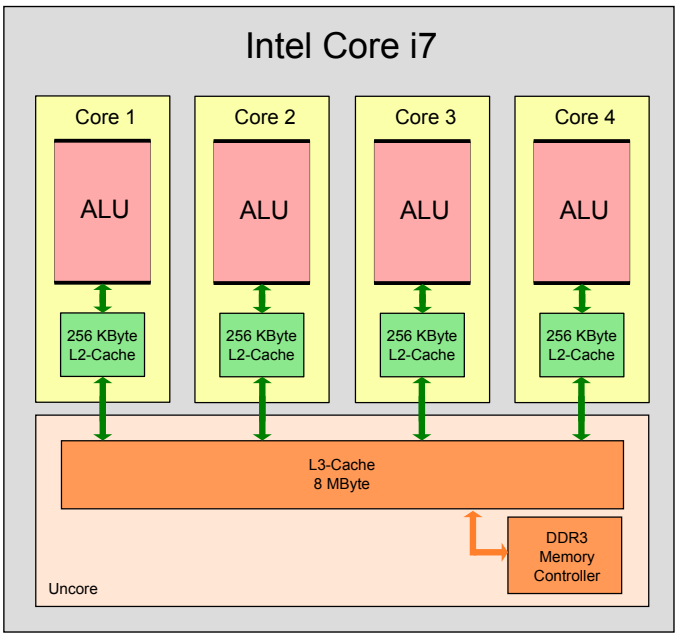


Abbildung 3.5.: Intel Core i7 Prozessor mit 4 Kernen. Jeder Kern verfügt über 256 kB L2-Cache. Alle Kerne werden über den L3-Cache mit einer Größe von 8 MB über einen DDR3 Speichercontroller an den Hauptspeicher angebunden. Der Bereich bestehend aus L3-Cache und Speichercontroller wird als „Uncore“ bezeichnet.

32-bit Architektur, da sie viele interne Strukturen besitzt, die selbstständig Optimierungen am Ablauf vornimmt. Die heute am Markt befindlichen Prozessoren besitzen zum Großteil eine „Out of Order“ Pipeline[2], was es ihnen ermöglicht, die Befehlsreihenfolge zu analysieren und umzusortieren, so dass ihre Rechenwerke gut ausgelastet sind. Eine hierarchische Cache Struktur sorgt für den transparenten und verzögerungsfreien Umgang mit Speicherzugriffen. Für genauere Einblicke in die Grundstrukturen sei auf [5] verwiesen.

Die Intel Nehalem Architektur [14] ist ein Vertreter für ein symmetrisches Multiprozessorsystem[3]. Kennzeichen dieser Hardwareklasse ist, dass jeder Rechenkern (Core) den gleichen Befehlssatz besitzt. Für einem Quadcore Prozessor ist das Design in Abbildung 3.5 dargestellt [15]. Im Vergleich zur vorherigen Strategie, bei dem der Speichercontroller über einen Bus extern angekoppelt wurde, ist bei der Nehalem Architektur der DDR3 Speichercontroller auf dem Prozessor mit integriert. Dadurch kann der Prozessor mit einer deutlich geringeren Latenzzeit auf den Speicher zugreifen als das bei Designs mit externem Speichercontroller möglich ist. Dies zeigt sich als Vorteil, da die

[2] In einer „Out of Order“ Pipeline können die Befehle durch den Prozessor selbst um sortiert werden. Ziel ist dabei eine höhere Auslastung der Recheneinheiten und somit eine höhere Leistung.

[3] Ein symmetrisches Multiprozessorsystem ist dadurch gekennzeichnet, dass alle Prozessoren über den selben Befehlssatz verfügen. Das Gegenteil hierzu ist ein asymmetrisches Multiprozessorsystem, wie es beispielsweise der Cell ist.

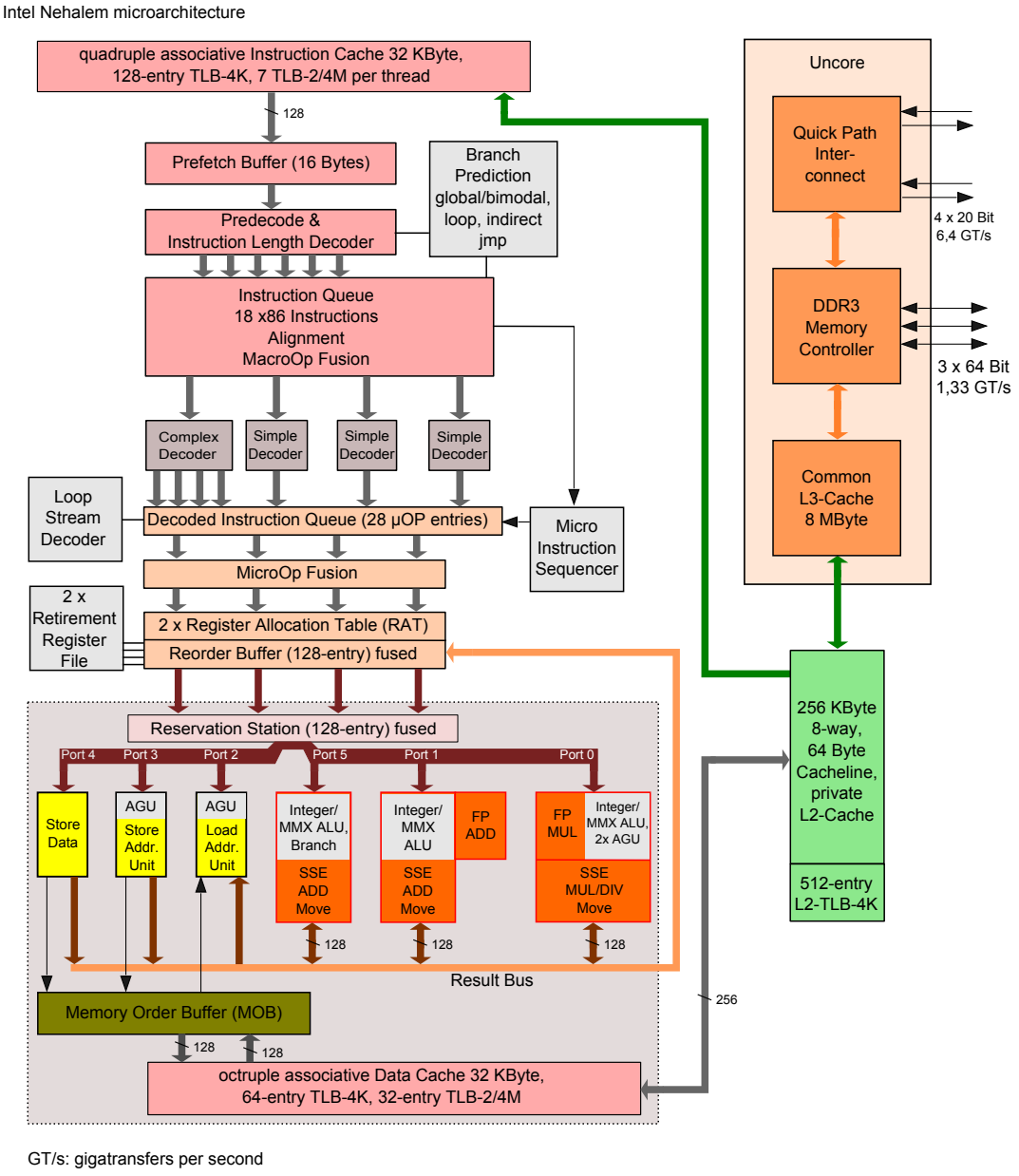


Abbildung 3.6.: Intel Nehalem Mikroarchitektur, wie sie in den Core i7 Prozessoren Verwendung findet.
Dieses Bild stammt aus der freien Enzyklopädie Wikipedia und steht unter der GNU-Lizenz für freie Dokumentation (Seite 85). Autor ist Appaloosa.

Rekonstruktionsalgorithmen in der CT meist durch die Speichertransferrate (siehe auch Kapitel 3.2) begrenzt sind. Daneben hat jeder Prozessor drei Speicherkanäle zur Verfügung, was einen spürbar höheren Durchsatz im Vergleich zur Vorgängergeneration bedeutet.

Für den Zugriff auf den Speicher sind noch mehrere Cache[4] Stufen zwischengeschaltet, um die für einen Prozessor langen Latenzzeiten zu verringern. Dabei sind jedem Kern 32 kB L1-Cache und 256 kB L2-Cache zur Verfügung gestellt. Alle Kerne teilen sich einen gemeinsamen L3-Cache, der je nach Prozessortyp zwischen 4 und 8 MB beträgt.

Ein für die Programmierung wichtiger Aspekt ist, dass sich in einem Prozessorkern mehrere Recheneinheiten (siehe Abbildung 3.6, Port 0–5) befinden. Bei näherer Betrachtung ist zu erkennen, dass hiervon drei pro Kern vorgesehen sind, die allerdings unterschiedliche Funktionen haben. Eine Einheit kann nur ganze Zahlen verarbeiten, die anderen beiden können auch mit Fließkommazahlen umgehen, wobei zu beachten ist, dass hiervon eine addieren und die andere multiplizieren kann. Auf diese Einheiten werden die ankommenden Instruktionen aufgeteilt, sofern dies wegen der Abhängigkeiten möglich ist. Bei einer geschickten Programmierung und dem entsprechenden Compiler ist es möglich, die Einheiten effizient aus zu lasten.

Bereits durch das Diagramm in Abbildung 3.6 wird klar, dass diese Architektur relativ komplex ist und viele Strukturen im Grunde nicht unbedingt notwendig wären, wenn der Code an sich besser auf die Architektur angepasst ist. Diesen Weg geht Intel mit neuen Designs, wie dem Larrabee. Der bereits auf dem Markt verfügbare Cell Prozessor verfolgt ebenso dieses Designziel.

3.3.2. Cell Broadband Engine

Die Cell Broadband Engine wurde durch ein Konsortium aus Sony, Toshiba und IBM entwickelt. Das Entwicklungsziel war einen kostengünstigen, aber dennoch leistungsfähigen Prozessor für Spiele, Multimedia und andere Anwendungen zu entwickeln. Vom grundsätzlichen Aufbau ist der Cell Prozessors ein asymmetrisches Multiprozessorsystem. Der interne Aufbau ist in Abbildung 3.7 zu finden. Der Prozessor besteht aus einem vollwertigen 64-bit PowerPC Prozessorkern (PPE) und acht kleineren, aber spezialisierten und optimierten Vektorprozessoren (SPEs). Daneben sind noch weitere Einheiten integriert, die für den Speicherzugriff und die Verbindung zu weiterer Peripherie zuständig sind. Die Elemente sind über einen Bus (EIB) miteinander verbunden.

[4]Ein Cache ist ein transparentes Element, das einen eigenen Speicher besitzt. Bei einer Anfrage versucht er Daten aus seinem Speicher zurück zu liefern, wenn dies nicht möglich ist, werden diese von der nächsten Stufe angefordert und zwischengespeichert.

Die hohe Leistung, die der Cell Prozessor erreichen kann, verlangt nach einer geschickten Aufteilung des Problems. Dabei muss jede Einheit und ihre Spezialitäten optimal ausgenutzt werden. Die PPE übernimmt dabei die Steuerung, da hierauf auch das Betriebssystem ausgeführt wird, die SPEs übernehmen dabei die Rolle der Rechenknechte und arbeiten die eigentliche Aufgabe ab. Daher ist die SPE für uns von besonderer Interesse, der ausführbare Code wird besonders für die SPE optimiert. In Abbildung 3.8 ist die Struktur der SPE dargestellt. Die SPE hat 256 kB Local Store Speicher, der sowohl das Programm, wie auch die Daten bereithält. Dieser wird durch die DMA-Engine[5] mit Daten befüllt. Dies kann parallel zur Ausführung geschehen. Für die Ausführung steht der SPE ein Registerfile aus 128 Vektorregistern mit einer Breite von 128 Bit zur Verfügung, wobei sich ein Vektorregister aus vier Gleitkommawerten mit einfacher Genauigkeit zusammensetzt. Die SPE besitzt zwei verarbeitende Einheiten, die jeweils eigenständige Befehle ausführen können. Die Vector Permute Unit übernimmt das Laden der Register aus dem Local Store ebenso wie Befehle zum Vertauschen einzelner Vektoreinträge, dem so genannten Shuffleing[6]. Die eigentliche Rechenaufgabe übernimmt die Vector ALU Unit[7]. Diese kann Befehle mit bis zu drei Eingangsoperanden gleichzeitig verarbeiten und das Ergebnis wieder in ein Register zurückschreiben. Sie ist beim Cell der ersten Generation auf Fließkommazahlen einfacher Genauigkeit optimiert. Der wohl wichtigste Befehl für unsere Aufgaben ist der Multiply-Add Befehl, der in einem Taktzyklus die Operation $E = O1 \cdot O2 + O3$ ausführen kann.

3.3.3. Grafikprozessoren

Die auf Grafikberechnung optimierte Graphical Processing Unit (GPU) wird im wissenschaftlichen Bereich zunehmend auch zum Beschleunigen von Algorithmen eingesetzt. Dies war bis vor einiger Zeit jedoch nur auf Basis der zur Verfügung gestellten Schnittstelle für Grafikanwendungen möglich, weswegen die Anpassung von Algorithmen und die Implementierung auf der GPU sehr aufwändig war. Beispiele aus dem wissenschaftlichen Bereich sind in [33] zu finden. Eine dieser Schnittstellen ist beispielsweise OpenGL [37]. Geändert hat sich dies mit neuen Ansätzen, die die Grafikhardware direkt

[5] Unter DMA(Direct Memory Access) wird eine Speicherzugriffsart verstanden, die direkt über den Bus auf den Speicher zugreift. Dabei ist es möglich, ohne mithilfe der CPU einen Speicherteil zu transferieren bzw. zu kopieren.

[6] Als Shuffle wird das Vertauschen von Einträgen im Vektor bezeichnet. Oftmals wird hier durch eine Vorschrift aus zwei Vektoren ein neuer zusammengesetzt, wobei je nach Hardware nicht unbedingt alle Kombinationen möglich sind: Z.B. Vektor A(A1, A2, A3, A4) und B(B1, B2, B3, B4) wird zu C(B3, A2, A4, B1).

[7] Die Arithmetic Logic Unit (ALU) ist in einem Prozessor für die Ausführung von einfachen Berechnungen und Logikoperationen zuständig.

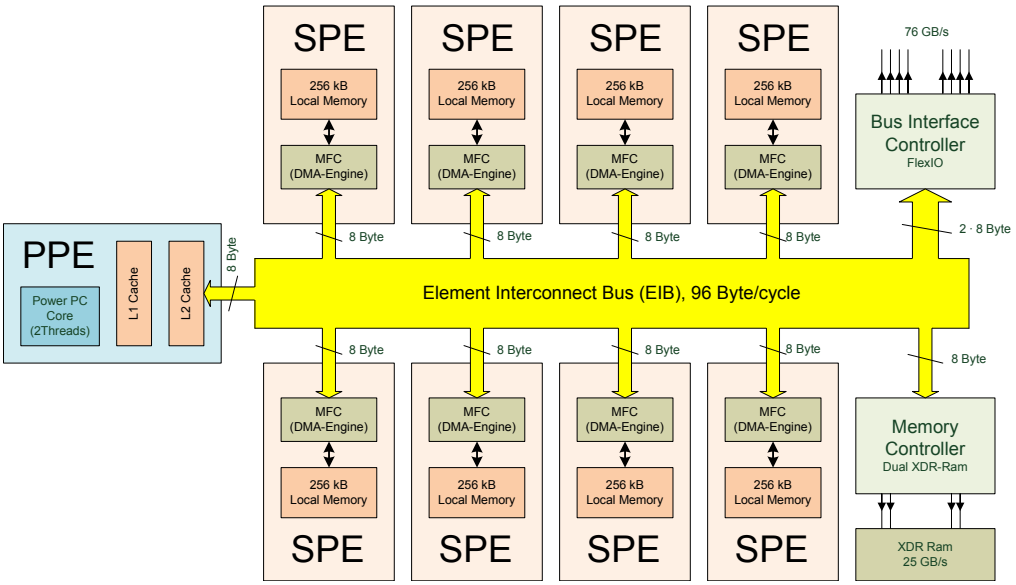


Abbildung 3.7.: Überblick über die Cell Architektur. Sie zeichnet sich durch 8 SPE (Synergistic Processing Elements) und dem Steuerkern, dem PPE (PowerPC Processing Element) aus. Untereinander sind diese durch den EIB verbunden.

ansprechen und programmieren können. Beispiele sind hierfür das von NVIDIA vorgestellte „Compute Unified Device Architecture“ (CUDA) [32] oder „Open Computing Language“ (OpenCL) [30]. Mit CUDA hat NVIDIA eine proprietäre und herstellerspezifische Sprache geschaffen. Der von NVIDIA zur Verfügung gestellte Compiler erzeugt den Code für die Ausführung auf der NVIDIA GPU.

Bei den GPUs dominieren momentan die Hersteller NVIDIA und AMD (mit der Marke ATI) den Markt. Die Grundkonzepte für beide GPUs sind dabei ähnlich, im Folgenden wird jedoch nur auf die Hardware von NVIDIA eingegangen, da hier die Struktur bekannt ist und mit CUDA eine einfach zu handhabende Programmierschnittstelle zur Verfügung steht. Die GPU, wie sie in Abbildung 3.9 dargestellt ist, besteht aus mehreren Multiprozessoren, auf denen jeweils unterschiedlicher Code ausgeführt werden kann. Ein Multiprozessor kann immer nur den gleichen Code ausführen, jeder interne Prozessor führt dabei den gleichen Befehl auf unterschiedliche Daten aus. Um die Prozessoren mit Daten zu versorgen, steht ihm ein Constant Cache (hier sind nur Lesebefehle auf einen speziellen Speicherbereich mit einer Größe von 64 kByte) und ein Texture Cache (hier sind Lesebefehle über das gesamte Device Memory möglich) zur Verfügung. Das Shared Memory spielt eine besondere Rolle, es ist ein schreib- wie lesbarer Speicher, auf den von den Prozessoren aus sehr schnell zugegriffen werden kann. Daneben sind noch direkte Zugriffe auf das Device Memory möglich. Diese sind jedoch wegen des fehlenden Caches vergleichsweise langsam. Bei der Algorithmenentwicklung für die GPU muss die Architektur besonders berücksichtigt werden. Es müssen möglichst viele unabhängig

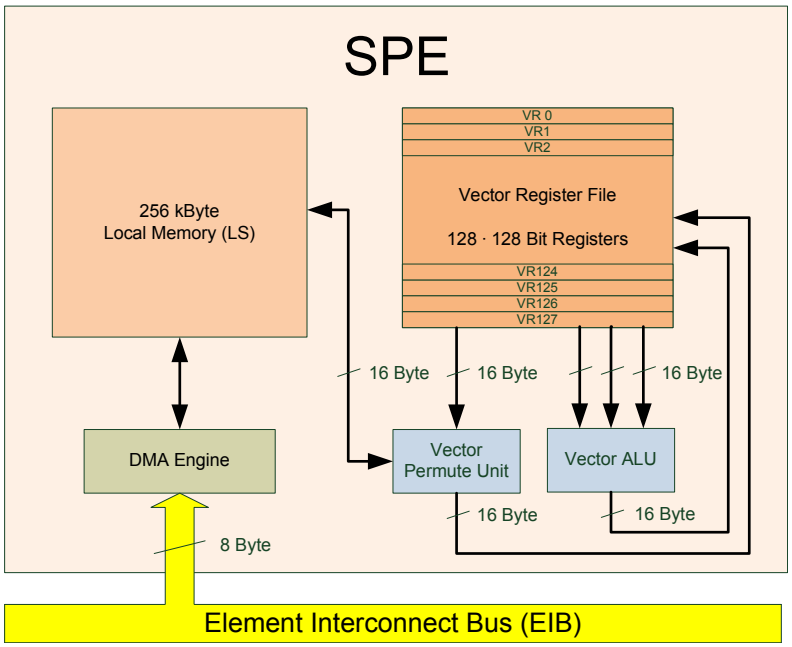


Abbildung 3.8.: SPE Element. Es besteht aus dem als Local Store (LS) bezeichnete Speicher, dem DMA Controller, um Daten mit dem Hauptspeicher auszutauschen, die 128 Vektorregister und der Rechenwerke. Die ALU ist die Recheneinheit, die Permute Unit ist für das Laden und Speichern im LS, sowie das vertauschen von Vektoreinträgen zuständig.

voneinander laufende Threads erstellt werden können, denn nur diese ermöglichen es, die Prozessoren gut auszulasten und die Speicherlatenz vernachlässigbar zu machen.

3.3.4. Vergleich der Architekturen

Alleine aus dem grundsätzlichen Aufbau lässt sich eine erste Eignung der Hardware für CT Rekonstruktionsalgorithmen ziehen. Das Problem, wenn verschiedene Architekturen verglichen werden sollen, ist jedoch, dass diese oftmals einen anderen Aufbau haben. Da das Rekonstruktionsproblem hauptsächlich vom vorhandenen Speicher, dessen hierarchische Struktur und Transferrate abhängig ist, wird nun im folgenden ein Modell entwickelt, auf das sich sowohl die x86 und die Cell-Architektur abbilden lässt. Ferner lassen sich darauf auch Architekturen, wie beispielsweise eine GPU abbilden. Damit lässt sich ein allgemeines Modell für die Hardwarearchitektur entwickeln (Abbildung 3.10). Der Kern der Architekturen ist das Rechenwerk (Processing Unit, PU). Hier werden die Befehle abgearbeitet. Zum Abarbeiten sind jedoch Daten erforderlich, die bereitgestellt werden müssen. Wie dies erfolgt, darin unterscheiden sich die Architekturen grundlegend. Zusammenfassen lässt sich dies jedoch immer in einem Speicher, der nahe an der PU liegt und Speicher, der weiter entfernt von der PU liegt. Die letzte Instanz ist dann noch der Hauptspeicher, der am langsamsten, aber auch am größten ist.

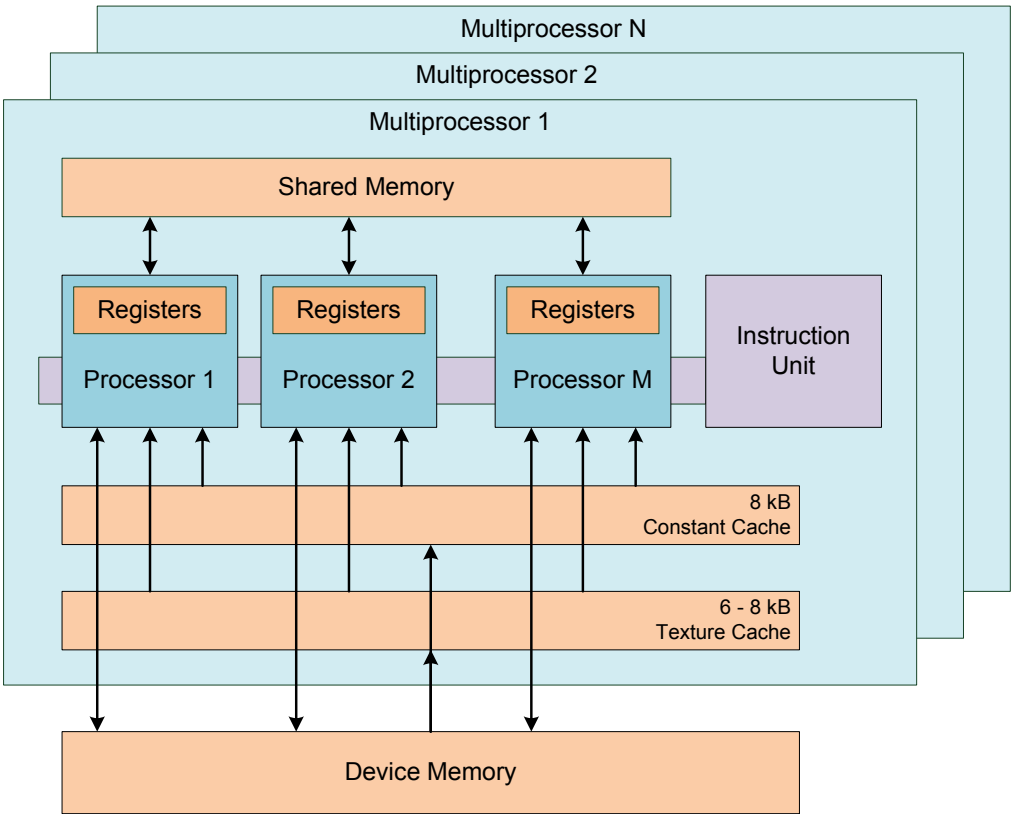


Abbildung 3.9.: Überblick über die Architektur einer CUDA fähigen NVIDIA GPU.

Die einzelnen speichernden Bestandteile lassen sich neben der Größe noch durch die Transferrate, mit der sie Daten liefern können und der Latenzzeit, also die Zeit, bis sie die ersten Daten liefern, beschreiben. Diese Größen wurden von den Herstellern entsprechend der erwarteten Einsatzbereiche ausgelegt. Grundsätzlich lässt sich aussagen, dass näher an der PU liegende Elemente einen kleinen Speicher, der eine geringe Latenz und einen hohen Durchsatz liefert, besitzen.

Eingesetzte Hardware

x86 Architektur (PC-Architektur) Als Vertreter der x86 Architektur wird die neuste Microarchitektur mit dem Codenamen Nehalem von Intel verwendet. Als CPUs kommen dabei der Intel Xeon X5570, 2,93 GHz mit dem Codenamen Gainestown zum Einsatz. Diese CPU zeichnet sich durch 4 Kerne aus, die jeweils 32 kB L1-Cache, 256 kB L2-Cache und pro Prozessor zusätzlich noch 8 MB L3-Cache besitzen. Daneben haben sie eine schnelle Anbindung an DDR3 Speicher, der über 3 Speicherkanäle angesprochen wird.

Cell Architektur Der Dual-Cell-Blade mit den zwei Cell Prozessoren, der mit 3,2 GHz getaktet ist, stammt von der Firma Mercury Computer Systems, Berlin, Deutsch-

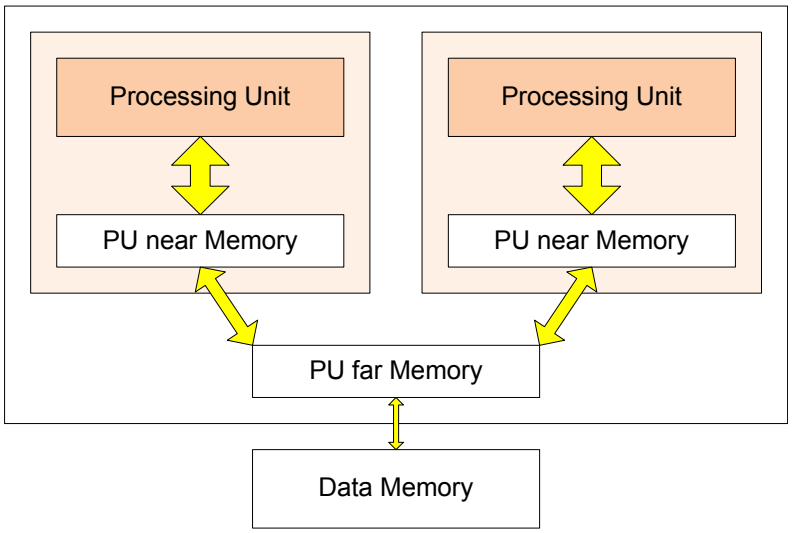


Abbildung 3.10.: Hardwaremodell zum Vergleich verschiedener Architekturen untereinander hinsichtlich der Eignung für CT Rekonstruktionsalgorithmen.

land. Als Hauptspeicher steht dem System ein 2 GB großes XDR-RAM zur Verfügung.

GPU Für die GPU wird der NVIDIA GTX 280 Prozessor untersucht. Der GPU standen 1 GB DDR3 Speicher zur Verfügung.

In Tabelle 3.3 sind die charakteristischen Werte beider Systeme zusammen gefasst. Die Werte für den Cell und die SPE stammen aus theoretischen Berechnungen und der vorhandenen Dokumentation. Die Werte für die SPE lassen sich komplett deterministisch vorhersagen, der theoretische Datentransfer für das XDR-Ram wurde durch die Anwendung in der Praxis bestätigt. Die Werte für die Intel Nehalem Architektur wurden durch Messungen am Testsystem mit dem Analysetool „Rightmark Memory Analyser“ in der Version 3.80 (http://cpu.rightmark.org/) gewonnen. Die Werte für die GPU wurden zum einen aus dem CUDA-Manual [32] entnommen oder durch Benchmarks ermittelt.

Speichertransferrate und -strukturen

Bei der Analyse ist zu erkennen, dass die SPE im Cell Prozessor von der Datenrate äquivalent zum L1-Cache der Nehalem Architektur ist. Beim Cell sind jedoch doppelt so viele Rechenkerne vorhanden, wodurch sich Datenrate beim Cell auf $381\,\mathrm{GB/s}$ und beim Intel auf $175\,\mathrm{GB/s}$ für den L1-Cache beläuft. Der L1-Cache ist für die meisten Algorithmen zu klein, weswegen oftmals auf den L2-Cache zurückgegriffen werden muss. Dieser hat eine akkumulierte Durchsatz von $116\,\mathrm{GB/s}$ pro Prozessor. Bei der GPU ist die Betrachtung vom gewählten Speichertyp abhängig. Der L1-Texture Cache sowie das

Tabelle 3.3.: Modellelemente für die Architekturen

Architekturelement		Cell Prozessor	Intel X5570		NVIDIA GTX 280	
Processing Unit		SPE	Core		Streaming Multiprocessors (SM)	
	typ. Prozessorenanzahl	2	2		1	
	Prozessorkerne	8	4		30	
	Vektorlänge	4	4		8	
	Taktrate	3,2 GHz	2,93 GHz		1,3 GHz	
	Peak Performance	105 GFlops	94 GFlops		933 GFlops	
PU near memory		Local Store	L1-Cache	L2-Cache	L1-Texture-Cache	Shared-Memory
	Größe	256 kByte	32 kByte	256 kByte	6 kByte	16 kByte
	Latenz	6 cycles	4 cycles	8–20 cycles	306 cycles	1 cycle
	Durchsatz	16 B/cycle	16 B/cycle	10,6 B/cycle	$4 \cdot 8$ B/cycle	$4 \cdot 8$ B/cycle
	Transferrate	47,7 GB/s	43,7 GB/s	28,9 GB/s	38,7 GB/s	38,7 GB/s
PU far memory		—	L3-Cache		L2-Texture-Cache	
	Größe	—	8 MByte		256 kByte	
	Latenz	—	20–70 cycles		$\approx$ 400 cycles	
	Durchsatz	—	8 B/cycle			
	Transferrate	—	21,8 GB/s		10 bis 13 GB/s	
Main memory	Technologie	XDR	DDR3		GDDR3	
	Latenz	$\approx$ 400 cycles	$\approx$ 200 cycles		400 bis 600 cycles	
	Durchsatz	8 B/cycle	4 B/cycle		64 B/cycle	
	Transferrate	23,8 GB/s	10,9 GB/s		141,7 GB/s	
	Transferrate pro Kern	3,0 GB/s	2,7 GB/s		4,7 GB/s	

Shared-Memory verfügt über eine akkumulierte Durchsatz von $1161\,\mathrm{GB/s}$. Diese ist auf den ersten Blick sehr beeindruckend, kann in der Praxis jedoch nur seltenst erreicht werden, da die Größen des Caches wie auch des Shared-Memories für die meisten Probleme zu gering sind. Wenn die GPU darüber hinaus den L2-Cache nutzt, sind noch ca. $390\,\mathrm{GB/s}$ verfügbar. Dabei ist jedoch zu beachten, dass auf der GPU wesentlich mehr Threads (typischerweise 512 oder mehr) gleichzeitig laufen und diese sich untereinander den Cache teilen.

Mit der maximalen Datenrate von $141{,}7\,\mathrm{GB/s}$ ist die GPU einsamer Spitzenreiter, was die Transferleistung aus dem Hauptspeicher angeht. Danach folgt der Cell mit seinem XDR Speicher und $23{,}8\,\mathrm{GB/s}$. Dieser ist klar im Vorteil gegenüber der Intel Nehalem Architektur, die hier mit $10{,}9\,\mathrm{GB/s}$ hinten liegt, auch wenn sich im Vergleich zum letzten Architekturschritt der Datenrate um den Faktor drei erhöht hat. Bei dieser Betrachtung dürfen allerdings weitere Faktoren, wie die Kernanzahl nicht außer Acht gelassen werden. Der Cell muss von dieser Speichertransferrate 8 Kerne versorgen, dahingegen hat die Intel Nehalem Architektur hier nur 4 zu versorgen. Daraus lässt sich das in der Tabelle 3.3 angegebene, bereinigte Maß der Speichertransferrate pro Kern ableiten. Hier liegt die Intel Nehalem Architektur knapp hinter dem Cell, die GPU hat hier eine um 60% höhere Transferrate zur Verfügung.

Speicherlatenzzeiten

Die in der Tabelle 3.3 angeführten Latenzzeiten auf dem Cell spielen in der Praxis so gut wie keine Rolle, das es dank der 128 Vektorregister (zum Vergleich, Intel X5570 hat nur 16 Vektorregister) einfach möglich ist, die Daten lange genug im Voraus zu laden und somit diese Zeiten zu überbrücken. Anders hingegen sieht dies bei der Intel Nehalem Architektur aus. Hier ist die Latenz für den L1-Cache sehr niedrig, diese steigt jedoch wesentlich an, wenn der L2-Cache mit betrachtet wird. Die Gesamtlatenz für die ersten 256 kB beträgt hier zwischen 8 und 20 Prozessortakten. In der Praxis senken weitere Mechanismen, wie beispielsweise Software- und Hardware Prefetching diese Zeiten noch deutlich, allerdings sind diese Mechanismen, insbesondere wenn sie in Hardware implementiert sind, undurchsichtig und schwer vorher zusagen. Hier lässt sich der Cell mit seiner unabhängig arbeitenden SPE sehr gut beschreiben und daher auch optimieren. Die Intel Nehalem Architektur spielt hingegen ihre Stärke bei großen Speicherbereichen mit verteilten Zugriffen besonders aus. Hier hilft die unkomplizierte Handhabung der Cache-Strukturen; selbst mit geringem Aufwand ist eine hohe Leistung zu erreichen. Sobald auf dem Cell mehr Daten auf einmal benötigt werden, als im Local Store Platz finden, müssen diese durch DMA-Transfers aus dem Hauptspeicher

nachgeladen werden, was eine hohe Latenzzeit nach sich zieht. Dies ist jedoch nur rentabel, wenn die Daten in größeren Blöcken sind und oftmals benutzt werden. Die GPU verfolgt hier einen anderen Ansatz, um die Latenzzeiten zu überbrücken. Hier werden so viele Threads angelegt wie für das Überbrücken der Latenzzeit notwendig sind. Dies setzt natürlich voraus, dass sich das Problem hocheffizient aufteilen lässt.

Rechenleistung

Neben dem Speichertransfer und der Latenzzeit ist auch die Rechenleistung ein wichtiges Kriterium. Für uns ist hier wichtig, dass besonders die Gleitkommarechnung mit einfacher Genauigkeit (float) eine sehr hohe Leistung liefert. Von den Herstellern wird oftmals ein theoretischer Wert der Peak-Performance angegeben, der jedoch in der Praxis kaum zu erreichen ist. Wie nahe die tatsächliche Leistung an das theoretische Maximum heranreicht, ist dabei von einigen Faktoren abhängig. Dazu zählt z.B. die effiziente Umsetzbarkeit des Codes auf die Befehle. Hierbei spielen der Cell und die Intel Nehalem Architektur ihre Vorteile nur aus, wenn mit Vektorbefehlen gearbeitet wird. Bei der GPU ist dies ein wenig anders, denn sie ist vom äußeren Erscheinungsbild ein skalarer Prozessor. Jedoch muss auf allen Prozessoren eines Multiprozessors immer den gleichen Befehl während eines Zykluses ausführen werden. Er stellt sich damit als ein flexiblerer Vektorprozessor da, da die Vektoren mit den Daten nicht linear im Speicher angeordnet sein müssen, dies aber für die Performance vorteilhaft ist. Daneben können die Intel Nehalem Architektur und der Cell durch ihre Superskalarität auch mehrere Befehle parallel ausführen, so z.B. neben einer Berechnung einen Wert aus dem Speicher laden. Dies ist bei der GPU nicht möglich, weswegen das Laden ein ganz normaler Befehl ist und somit einen Taktzyklus benötigt. Ein weiterführender Vergleich zwischen GPU und den beiden anderen Architekturen ist nur anhand von Beispielen möglich. Die erbrachte Leistung der GPU ist stark vom Problem abhängig.

Fazit aus diesem Vergleich

Der Unterschied zwischen dem Cell Prozessor und der neue Intel Nehalem Architektur ist nicht sonderlich groß, wenn die Resultate aus dem Vergleich herangezogen werden. Von der Leistung ist also ein sehr ähnliches Verhalten zu erwarten, wenn auch der Cell-Prozessor durch sein Design komplexer in der Programmierung ist. Die Grafikkarte hingegen hat auf den ersten Blick große Vorteile: Die große Speichertransferrate und die hohe Rechenleistung. Jedoch stehen diesem auch große Nachteile gegenüber bzw. die Aussagen müssen relativiert werden. Die gesamte Rechenleistung lässt sich

nur dann sinnvoll nutzen, wenn das Problem sehr gut zu parallelisieren ist und der Algorithmus gut auf die GPU angepasst werden kann. Je mehr Rechenleistung im Vergleich zum Speichertransfer benötigt wird, umso effizienter wird die Architektur. Dies ist jedoch bei den Algorithmen in der CT nicht unbedingt gegeben, da diese durch die Speichertransferrate begrenzt sind. Während dann die Intel Nehalem wie auch die Cell Architektur aus ihrem PU near memory schöpfen können, ist dies bei der GPU aufgrund der kleinen Größe nicht möglich und es muss dazu der mit weniger Transferrate angebundene Hauptspeicher verwendet werden. Diese These wird durch Beobachtungen aus Messungen für verschiedene Rückprojektionsalgorithmen gestützt. Diese sind in der Tabelle 5.3 auf Seite 69 zu finden. Dabei ist zu erkennen, das je geringer der Rechenaufwand für Geometrieberechnungen wird, umso weniger Unterschied ergibt sich zwischen den GPUs und den CPUs. Dies ist in der Gegenüberstellung von der perspektivischen Rückprojektion gegenüber der weniger rechenaufwändigen parallelen Rückprojektion zu finden. Aus diesem Grund wurde auch darauf verzichtet, den Algorithmus auf der GPU zu implementieren.

3.4. Optimierung und Anpassung der Rückprojektion an die Architektur

Bereits im Kapitel 3.2 wird die Optimierung durch geometrische Berechnungen beschrieben. Dies ist ein Schritt zum optimierten Algorithmus. Der dargestellte Referenzcode aus Listing 2 ist hinsichtlich der Optimierung jedoch noch nicht ausgereizt, denn die Referenzimplementierung ist noch wenig an die Hardware-Architektur angepasst. Auch hier hat der Leitfaden aus [25] noch einige Tricks zu bieten, wie eine bessere Anpassung an die Architektur erreicht werden kann.

Optimierung der Nutzung des Caches

Auf der x86 Architektur ist ein essenzieller Schritt die Anpassung des Programms an den Cache und dessen Struktur. Auf dem Cell Prozessor sind ähnliche Schritte notwendig, um die Daten in den Local Store zu laden, weswegen diese Überlegungen sehr ähnlich sind.

Wie bereits im Kapitel 3.2 dargelegt wurde, müssen von den Speichern zu den Rechenwerken lesend die doppelte Menge an $\mathrm{GU} \cdot$ `sizeof(float)` und schreibend die Menge an $\mathrm{GU} \cdot$ `sizeof(float)` transferiert werden. Ein typischer Spiralscan, wie er

heute klinisch verwendet wird, hat bei einem Volumen mit 512^3 Voxeln einen Aufwand von $48{,}2\,\mathrm{GU}$. Als Flaschenhals erweist sich dabei zunächst der lesende Zugriff auf den Speicher mit $386\,\mathrm{GB}$. Werden hierbei die Transferraten aus dem Modell in Tabelle 3.3 zugrunde gelegt, ist bei einem Zugriff aus dem Main Memory mit einer Rekonstruktionszeit von $35\,\mathrm{s}$ zu rechnen. Wenn hingegen die Daten im L3-Cache vorliegen, dann ist mit einer Zeit von $18\,\mathrm{s}$ zu rechnen. Eine deutliche Steigerung ergibt sich durch den wenig schnelleren, aber dafür in eine Prozessor 4 mal vorhandenen L2-Cache auf eine theoretische Zeit von $3\,\mathrm{s}$.

Die Cache-Nutzung des Referenzcodes ist schlecht, da eine zu große, verteilte Datenmenge auf einmal verwendet wird. Um die Effizienz zu erhöhen, muss der Algorithmus so verändert werden, dass eine höhere Datenlokalität entsteht. Es muss sich ein Zugriffsmuster auf die Rohdaten ergeben, das es erlaubt, diese in dicht aufeinander folgenden Zugriffen wieder zu verwendet. Dies kann bei einer Rückprojektion sehr einfach erreicht werden, indem aus dem großen Volumen ein kleines Volumen herausgeschnitten wird. Es wir dann bei der Projektion nur der Bereich verwendet, der dem „Schatten“ des Volumens entsprechen würde. Weiter lässt sich die Lokalität beim Volumen erhöhen, indem mehrere Projektionen (die aus benachbarten Winkeln aufgenommen wurden) in diesen Prozess mit einbezogen werden.

Weitere Anpassung an die Architektur

Bei den modernen Architekturen, wie der Intel Nehalem und dem Cell, wird die volle Leistung nur erreicht, wenn auch die Vektoreinheiten verwendet werden. Dies ist jedoch in der in Kapitel 3.1.4 vorgestellten Rückprojektion nur sehr ineffizient möglich, da hier keine direkten Vektoren gebildet werden können. Vielmehr müssen diese zunächst zusammengesetzt werden. Gut zu erkennen ist dies im Listing 2 auf Seite 15. Wird hier die Zeile `V(i, j, k) += w * R(l, m, n); // update Voxel` und der Aufbau des Volumens betrachtet, so ist zu erkennen, dass zwar das Volumen (für uns die wichtige `k`-Richtung) als Vektor gelesen werden können, da diese nacheinander benötigt werden. Allerdings müssen die verwendeten Rohdaten zunächst in diese Vektor-Form umsortiert werden, da jeder Rohdatenzugriff `R(l, m, n)` auf eine andere Rohdatenstelle zugreift, die nicht zwangsläufig hintereinander im Speicher angeordnet ist. Dies macht die Vektorisierung sehr ineffizient, einen deutlichen Fortschritt bei der Spirale ist hier die Implementierung mithilfe von gedrehten Schichten, die so angepasst wurde, dass die Nutzung von sehr langen Vektoren möglich wird. Diese Möglichkeit der Lösung wird im nächsten Kapitel genauer vorgestellt.

4. Schnelle Implementierung durch gedrehte Schichten

Für die heute verfügbaren Rechner ist eine gute Vektorisierung Pflicht, wenn eine hohe Performance erreicht werden soll. Das Problem der Vektorisierung lässt sich jedoch bei der Rückprojektion wie bereits gezeigt nicht trivial lösen. Für die Spiral-CT besteht eine Möglichkeit darin, die vorhandene Symmetrie zu nutzen. Ziel ist es dabei, eine neue Geometrie einzuführen, in der die Schichten so angeordnet sind, dass diese für die Gewichte und die Rohdaten-Lookups verschiebungsinvariant in der z-Richtung sind. Dafür muss folgende Bedingung gelten:

$$w(x, y, \alpha) = w(x, y, \alpha + n \cdot \Delta\alpha) \qquad \forall\, n \in \mathbb{R} \tag{4.1}$$

Der Parameter α ist dabei ein Wert, der einer z-Position im Volumen entspricht. Eine Lösung für diese Bedingung ist das Drehen der Schichten mit der Spirale. Dabei wird der Parameter α sowohl für die z-Koordinate, wie auch für die Drehung der Schichten verwendet. Die Schichten werden dabei so gedreht, dass die Quellentrajektorie die Schichten immer an der gleichen Stelle im neuen Koordinatensystem schneidet.

Dies hat zwei wichtige Auswirkungen auf den Prozess der Rückprojektion. Die neue Anordnung verkleinert die benötigten Lookup-Tabellen für die Rohdaten und die Gewichte um eine Dimension. Dadurch wird zum einen Speicherplatz gespart, zum anderen werden die notwendigen Berechnungen um die nun fehlende Dimension verringert. Als weiterer Vorteil ist der Algorithmus nun vektorisierbar, wenn die Rohdaten umsortiert werden. Dies führt zu einer speichereffizienten und schnellen Implementierung mit der Möglichkeit zur Verwendung von Lookup-Tabellen.

Die Details sind in folgenden Peer-Reviewed Veröffentlichungen zu finden, die im Rahmen der Arbeit entstanden und im Folgenden abgedruckt sind:

STECKMANN, S. ; KNAUP, M. ; KACHELRIESS, M. : High performance cone–beam spiral backprojection with voxel–specific weighting. In: *Phys. Med. Biol.* 54 (2009), S. 3691–3708

STECKMANN, S. ; KNAUP, M. ; KACHELRIESS, M. : Algorithm for hyperfast cone–beam spiral backprojection. In: *Computer Methods and Programs in Biomedicine* (In press, 2009)

IOP PUBLISHING PHYSICS IN MEDICINE AND BIOLOGY

Phys. Med. Biol. **54** (2009) 3691–3708 doi:10.1088/0031-9155/54/12/006

High performance cone-beam spiral backprojection with voxel-specific weighting

Sven Steckmann, Michael Knaup and Marc Kachelrieß

Institute of Medical Physics (IMP), University of Erlangen-Nürnberg, Henkestr. 91, 91052 Erlangen, Germany

E-mail: sven.steckmann@imp.uni-erlangen.de

Received 1 October 2008, in final form 24 April 2009
Published 28 May 2009
Online at stacks.iop.org/PMB/54/3691

Abstract

Cone-beam spiral backprojection is computationally highly demanding. At first sight, the backprojection requirements are similar to those of cone-beam backprojection from circular scans such as it is performed in the widely used Feldkamp algorithm. However, there is an additional complication: the illumination of each voxel, i.e. the range of angles the voxel is seen by the x-ray cone, is a complex function of the voxel position. In general, one needs to multiply a voxel-specific weight $w(x, y, z, \alpha)$ prior to adding a projection from angle α to a voxel at position x, y, z. Often, the weight function has no analytically closed form and must be numerically determined. Storage of the weights is prohibitive since the amount of memory required equals the number of voxels per spiral rotation times the number of projections a voxel receives contributions and therefore is in the order of up to 10^{12} floating point values for typical spiral scans. We propose a new algorithm that combines the spiral symmetry with the ability of today's 64 bit operating systems to store large amounts of precomputed weights, even above the 4 GB limit. Our trick is to backproject into slices that are rotated in the same manner as the spiral trajectory rotates. Using the spiral symmetry in this way allows one to exploit data-level paralellism and thereby to achieve a very high level of vectorization. An additional postprocessing step rotates these slices back to normal images. Our new backprojection algorithm achieves up to 17 giga voxel updates per second on our systems that are equipped with four standard Intel X7460 hexa core CPUs (Intel Xeon 7300 platform, 2.66 GHz, Intel Corporation). This equals the reconstruction of 344 images per second assuming that each slice consists of 512×512 pixels and receives contributions from 512 projections. Thereby, it is an order of magnitude faster than a highly optimized code that does not make use of the spiral symmetry. In its present version, the spiral backprojection algorithm is pixel-driven. A ray-driven version and a corresponding high performance forward projector can be easily designed. Thus, our findings can be used to speed up any type of image reconstruction algorithm (approximate

or exact analytical algorithms and iterative algorithms) and therefore yield a versatile and valuable component of future image reconstruction pipelines.

(Some figures in this article are in colour only in the electronic version)

1. Introduction

High performance image reconstruction (HPIR) basically means high performance backprojection since backprojection is the most demanding step in the image reconstruction pipeline. Recently, we published our work on high performance parallel beam backprojection and high performance perspective cone-beam backprojection using cell broadband engine (CBE) based implementations and central processing unit (CPU) based implementations (Kachelrieß *et al* 2007).

While our previous work focused on fast reconstruction of circular cone-beam and parallel-beam data, we are here interested in reconstructing spiral CT rawdata. In spiral CT, the patient is translated through the gantry while the x-ray source and the x-ray detector rotate around the patient (Kalender 2005). Thereby, the focal spot trajectory follows a helical or spiral path as opposed to the more simple circular trajectory. This is illustrated in figure 1.

The most important and the most computational demanding step in spiral CT image reconstruction is the backprojection step. It typically consumes about 90% of the total spiral CT image reconstruction time. Hence, the performance of spiral CT image reconstruction can significantly profit from an optimized spiral backprojection approach.

Spiral 3D backprojection is, however, more complicated than 3D backprojection from circle trajectories. The main reason is that the illumination of the voxels by the x-ray cone exhibits complex voxel location-dependent behavior that must be taken into account during the backprojection step. Basically, this requires to implicitly or explicitly compute a weight function $w(x, y, z, \alpha)$, where (x, y, z) is the voxel position and α is the projection angle of the ray that is backprojected, and to apply this weight to each voxel during the backprojection. Further on, the weight function may include the typical voxel-specific distance weighting, length correction weighting and other weighting factors necessary for modern spiral cone-beam CT image reconstruction. Numerous algorithms have been published that already use voxel-specific weighting (Kachelrieß *et al* 2004, Tang *et al* 2006, Taguchi *et al* 2004, Kudo *et al* 2003). Only then, each measured ray can contribute to the image. Other algorithms circumvent the problem of voxel-specific weighting by backprojecting data only from certain regions of the detector, such as data from the Tam window, for example (Danielsson *et al* 1998, Turbell 2001, Defrise *et al* 2000, Proksa *et al* 2000). This, however, implies a compromise in dose usage or image quality. Moreover, numerical reasons often enforce to avoid realizing such detector windows by masking the detector pixels and rather require the formulation as a voxel-specific weight. Similarly, implementations of quasiexact and exact reconstruction algorithms, such as those proposed in Bontus *et al* (2003), Katsevich (2002, 2006), Noo *et al* (2003), will profit from our new approach by just replacing their backprojection with that proposed in this paper and by formulating their detector window as a voxel-specific weight function.

This paper proposes a new spiral backprojection algorithm (Steckmann *et al* 2008a, Steckmann *et al* 2008b) that allows for voxel-specific weights and that can be helpful to boost the performance of most spiral image reconstruction algorithms, no matter whether these are approximate spiral cone-beam image reconstruction algorithms, exact spiral cone-beam

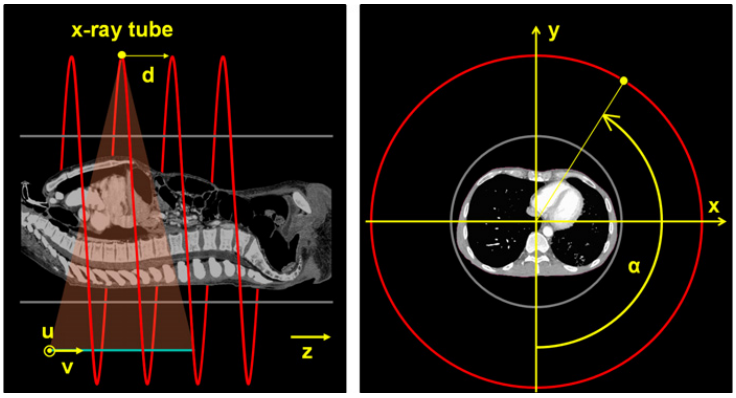


Figure 1. Illustration of the spiral trajectory. The lateral and longitudinal detector coordinates are u and v, respectively. The rotation angle is denoted as α. The table increment per rotation is denoted as d.

image reconstruction algorithms or are of iterative nature (in which case a corresponding forward projector must be programmed as well). Our algorithm reconstructs pixels from arbitrary detector shapes into arbitrary pixel grids; it is not fixed to the common Cartesian grid. Sophisticated representations such as the hexagonal grid are also feasible (Knaup *et al* 2007). Our spiral cone-beam CT backprojection algorithm does not work for circle scans.

This paper does not propose a new image reconstruction algorithm. Our focus is the high performance backprojection step and the use of symmetries which allow for highly performant implementations. Similar to the fast Fourier transform, which is just a tricky implementation of the discrete Fourier transform, our backprojection algorithm is just a tricky realization of backprojection. Both algorithms make strong use of symmetries. The fast Fourier transform assumes the transform length to be a power of 2 or to be at least factorizable into small integers. Our spiral backprojection assumes the scan trajectory to be a spiral trajectory. Just as the fast Fourier transform which cannot be used when the symmetry is broken, as is the case when the transform size is prime, our high performance spiral backprojection algorithm cannot be used when its assumptions are violated, e.g. when the spiral pitch value varies during the scan.

To evaluate our approach, we did implement a complete spiral CT reconstruction pipeline and we did perform complete image reconstructions.

2. Backprojection

Let α be the projection angle. The source position as a function of α is given as

$$\boldsymbol{s}(\alpha) = \begin{pmatrix} R_{\mathrm{F}} \sin\alpha \\ -R_{\mathrm{F}} \cos\alpha \\ đ\alpha \end{pmatrix}$$

where $đ = d/2\pi$ and d is the table increment per full rotation and R_{F} is the radius of the focal trajectory. Backprojection means evaluating

$$f(x, y, z) = \int \mathrm{d}\alpha \, w(x, y, z, \alpha) \hat{p}(\alpha, u, v),$$

with $u = u(x, y, z, \alpha)$ and $v = v(x, y, z, \alpha)$ being the detector's lateral and longitudinal coordinates respectively given by the intersection of the ray from $\boldsymbol{s}(\alpha)$ through the voxel

(x, y, z) with the detector surface. The preprocessed (and convolved if necessary) rawdata are denoted by $\hat{p}(\alpha, u, v)$.

It is difficult to compute the functions u, v and w efficiently. The computation of the detector coordinates u and v, which is the projection of voxel (x, y, z) from $\boldsymbol{s}(\alpha)$ onto the detector, typically involves the evaluation of rational functions or of trigonometric functions and thereby is time consuming. Even more difficult is the determination of the weight function w. As discussed in section 1, w comprises distance weighting and illumination normalization. Evaluating w typically involves a number of trigonometric function calls and often requires to numerically solve transcendental equations. To increase the performance of spiral backprojection, a lot of effort has to go into optimizing the evaluation of these three functions u, v and w, since this evaluation has to be done for every voxel and for every projection. Precomputing the values and storing them into look-up tables (luts) are prohibitive due to the large amount of memory required (each dimension x, y, z and α typically consists of 10^3 samples which implies that each of the three luts needs to hold in the order of 10^{12} samples).

Nevertheless, our aim is to precompute and store functions u, v and w and we will show below how to do so. It should be noted that storing these functions in luts is only an option to our backprojection algorithm. If u, v and w can be as well computed on the fly without impairing performance, one is free to do so. Whether the on-the-fly computation is to be preferred over luts or vice versa strongly depends on the spiral geometry used and on the image reconstruction algorithm.

Our primary aim is to vectorize the backprojection algorithm. Vectorization is essential for performance optimization since it minimizes the number of computer operations required. Further on, modern computer systems support vectorization with special single instruction multiple data (SIMD) commands. This means that an operation, such as addition or multiplication, does act not only on a single value but rather on a vector of typically four values. In its present form, the backprojection algorithm is not vectorized: for each value of x, y, z and α one has to load a rawdata value $\hat{p}$ (after computing u and v) and a voxel value f from memory into the processing unit, has to load or compute w and then has to compute $f + w\hat{p}$ and store the result into memory. A vectorized version would rather compute u, v and w once, and then load a vector of rawdata values $\hat{\boldsymbol{p}}$ and the same number of voxel values $\boldsymbol{f}$ and then weighted add those two vectors $\boldsymbol{f} + w\hat{\boldsymbol{p}}$. With increasing length of the vectors $\hat{\boldsymbol{p}}$ and $\hat{\boldsymbol{f}}$, the effort of computing u, v and w becomes more and more negligible. Further more, the weighted add of two vectors can be easily implemented using the modern SIMD commands (especially when the number of elements in $\hat{\boldsymbol{p}}$ and $\boldsymbol{f}$ is a multiple of 4), given that the vector elements are arranged consecutively in memory.

Now, we would like to illustrate the situation giving the reference code listing for the standard backprojection. To do so, we must introduce discrete variables as alternatives to the continuous variables used so far. Basically, the coordinates x, y and z are replaced by indices `i`, `j` and `k`, while the scanner coordinates v, u and α become `l`, `m` and `n`, respectively. The correspondences and the range of these indices are summarized in table 1. Listing 1 is the reference code for the standard spiral backprojection. Note that the innermost loop, i.e. the loop over index `k`, contains all the computations of u, v and w. After these have been determined only one scalar rawdata value is loaded, weighted and added to one voxel. This code is not vectorized and therefore will perform rather slow. A vectorization of this code is impossible since u, v and w differ from voxel to voxel.

Our idea of how to achieve full vectorization of the backprojection algorithm is to make use of the spiral symmetry. If we rotate each slice of the volume in the same manner the spiral trajectory rotates, then each slice requires the same computations before loading the rawdata

Table 1. Continuous and corresponding discrete variables.

Continuous	Discrete	Comment
x	`i`	`0<=i<I` where `I` is the number of voxels in x
y	`j`	`0<=j<J` where `J` is the number of voxels in y
z	`k`	`0<=k<K` where `K` is the number of voxels in z
v	`l`	`0<=l<L` where `L` is the number of detector rows
u	`m`	`0<=m<M` where `M` is the number of detector columns
α	`n`	`0<=n<N` where `N` is the number of projections contributing to one slice
f	`Vol(i,j,k)`	`Vol[(i*J+j)*K+k]`, linear layout with the z-direction being the fast index
p	`Raw`	data layout specified in the listings
$u(x, y, \alpha)$	`mLut(i,j,n)`	`mLut[(i*J+j)*N+n]` if stored as a look-up table and not computed on-the-fly
$v(x, y, \alpha)$	`lLut(i,j,n)`	`lLut[(i*J+j)*N+n]` if stored as a look-up table and not computed on-the-fly
$w(x, y, \alpha)$	`wLut(i,j,n)`	`wLut[(i*J+j)*N+n]` if stored as a look-up table and not computed on-the-fly

and adding them to the voxel. The computation of u, v and w thereby becomes independent of the slice's z-position which in turn reduces the memory requirements for the lut strategy by a factor of 10^3. Further on, if we arrange the voxel data and the rawdata in an elegant way we can achieve vectorization of the algorithm since we can reuse the same u, v and w for all slices.

To become more specific, let

$$\hat{f}(x, y, \alpha) = f(x\cos\alpha - y\sin\alpha, x\sin\alpha + y\cos\alpha, đ\alpha)$$

be a new representation of volume f. In the new volume $\hat{f}$, each slice rotates in the same fashion as the spiral trajectory. Hence, $\hat{f}$ exhibits the same spiral symmetry as the data acquisition itself. Due to the fact that the z-position in spiral CT is proportional to the gantry angle α, we were free to drop z in favor of α.

Now, backprojection is

$$\hat{f}(x, y, \alpha) = \int \mathrm{d}a\, w(x, y, a)\,\hat{p}(\alpha + a, u, v) \tag{1}$$

with $u = u(x, y, a)$ and $v = v(x, y, a)$. The angle a counts relative to the slice position α.

This new backprojection equation is much more favorable for implementation since neither the integrand's weight function nor the look-up values u and v depend on the absolute angle α (the z-position); they only depend on the relative angle a. Hence, we can loop over a range of z-positions α and add the corresponding projection entries to the volume without reevaluation of the weighting or look-up values. To illustrate this, let us rewrite equation (1) for consecutive slices where $\Delta\alpha$ shall represent the distance between the slices:

$$\begin{aligned}
\hat{f}(x, y, \alpha) &= \int \mathrm{d}a\, w(x, y, a)\,\hat{p}(\alpha + a, u, v) \\
\hat{f}(x, y, \alpha + \Delta\alpha) &= \int \mathrm{d}a\, w(x, y, a)\,\hat{p}(\alpha + \Delta\alpha + a, u, v) \\
\hat{f}(x, y, \alpha + 2\Delta\alpha) &= \int \mathrm{d}a\, w(x, y, a)\,\hat{p}(\alpha + 2\Delta\alpha + a, u, v) \\
\hat{f}(x, y, \alpha + 3\Delta\alpha) &= \int \mathrm{d}a\, w(x, y, a)\,\hat{p}(\alpha + 3\Delta\alpha + a, u, v) \\
&\vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots
\end{aligned}$$

We find that if we represent $\hat{f}$ and $\hat{p}$ with α being the linear (fast) variable in memory, we achieve to fully vectorize the backprojection. This can be easily seen if we have a look at

the above equations: they represent a vector of four (or more) elements. Note that functions w, u and v always evaluate to the same value for the whole vector. We have to calculate or look up these values only once for such a vector and we typically use vectors of 256 elements in our implementation (for a conventional spiral backprojection, this has to be done for every voxel in every slice).

Using the spiral symmetry removes the z-dependence of u, v and w, which are now no longer 4D functions, but rather 3D functions. Storing these in luts thus costs far less memory and can be easily done on modern PCs. As an alternative to using luts, one may also precompute u, v or w on the fly which also becomes less demanding since this computation only has to be carried out once per pixel and is then valid for all slices.

Listing 1: Reference standard backprojection algorithm.

```
void SpiralBP( int const I, // Number of x-pixels x
               int const J, // Number of y-pixels y
               int const K, // Number of slices   z
               int const L, // Number of rows     v
               int const M, // Number of channels u
               int const N, // Number of views a
             float const * const Raw,  // p(n, m, l)
             float       * const Vol)  // f(x, y, z)
     {
     #define V(i, j, k)  Vol[(i*J+j)*K+k]
     #define R(l, m, n)  Raw[L*M*n+L*m+l]

     for(int n=0; n<N; n++) // alpha-loop
     for(int i=0; i<I; i++) // x-loop
     for(int j=0; j<J; j++) // y-loop
     for(int k=0; k<K; k++) // z-loop
           {
             int const m=mCalc(n, i, j, k); // u(a, x, y, z)
             int const l=lCalc(n, i, j, k); // v(a, x, y, z)
           float const w=wCalc(n, i, j, k); // w(a, x, y, z)

           V(i, j, k)+=w*R(l, m, n); // update voxel value
           }

    #undef V
    #undef R
    }
```

The (non-optimized) reference source code of listing 2 illustrates the fast algorithm for backprojection. The listing also clearly shows why this algorithm is good for vectorization. The innermost loop, i.e. the loop over slices `k`, is the fastest index. For treating this loop as a vector, the data belonging to this loop must be arranged consecutively in memory. This can be seen from the data layout as specified by `#define` directives: the index `k` is a simple offset in the data and thus accesses are consecutively in memory.

Listing 2: Reference fast backprojection algorithm.

```
void SpiralBP( int const I, // Number of x-pixels x
               int const J, // Number of y-pixels y
               int const K, // Number of slices   z
               int const L, // Number of rows     v
               int const M, // Number of channels u
               int const N, // Number of views    a
               int const * const mLut, // u(x, y, a)
               int const * const lLut, // v(x, y, a)
```

```
        float const * const wLut, // w(x, y, a)
        float const * const Raw,  // p(alpha+a, u, v)
        float       * const Vol)  // f(x, y, alpha)
{
#define mlut(i, j, n) mLut[(i*J+j)*N+n]
#define llut(i, j, n) lLut[(i*J+j)*N+n]
#define wlut(i, j, n) wLut[(i*J+j)*N+n]
#define V(i, j, k)     Vol[(i*J+j)*K+k]
#define R(l, m, n, k)  Raw[((l*M+m)*(N+K)+n+k]

for(int i=0; i<I; i++) // x-loop
for(int j=0; j<J; j++) // y-loop
for(int n=0; n<N; n++) // alpha-loop
      {
        int const m=mlut(i, j, n); // u(x, y, a)
        int const l=llut(i, j, n); // v(x, y, a)
      float const w=wlut(i, j, n); // w(x, y, a)

      for(int k=0; k<K; k++) // z-loop
        V(i, j, k)+=w*R(l, m, n, k); // update voxel value
      }

#undef mlut
#undef llut
#undef wlut
#undef V
#undef R
}
```

The limitation of this version of the algorithm is that the distance of adjacent slices is fixed to the table increment Δz of the scanner between two adjacent projections (which may be rather tiny). This problem can be resolved by a new variable `dN` that counts how many multiples of Δz the slices shall be separated; the voxel size in the z-direction therefore is always a multiple of the table increment per projection. The extended algorithm is given in listing 3. Note that adding the freedom to specify the longitudinal sampling requires only the simple decomposition of the view number `n` into `n=kk*dN+dn`. This is nothing more than a reordering of the projections.

3. Slice rotation

The improved backprojection performance comes at the price of having the slices rotating together with the spiral as shown in figure 2. This requires us to add an additional rotation step to undo the inherent rotation and to finally end up with images in a standard Cartesian domain. This is an additional step in the reconstruction pipeline but it does not add to the total reconstruction time because slice rotation must be done only once at the end of the reconstruction while backprojection accesses each voxel about 10^3 times more often.

Another restriction of our fast spiral backprojection approach is that the field of view (FOV), i.e. the region within the field of measurement (FOM) that shall be reconstructed, must be centered in the isocenter, and it will be circular. To obtain rectangular FOVs, some clipping must occur after the rotation step unless the full FOM shall be reconstructed. Given the enormous improvements in reconstruction speed we achieve, this disadvantage becomes negligible.

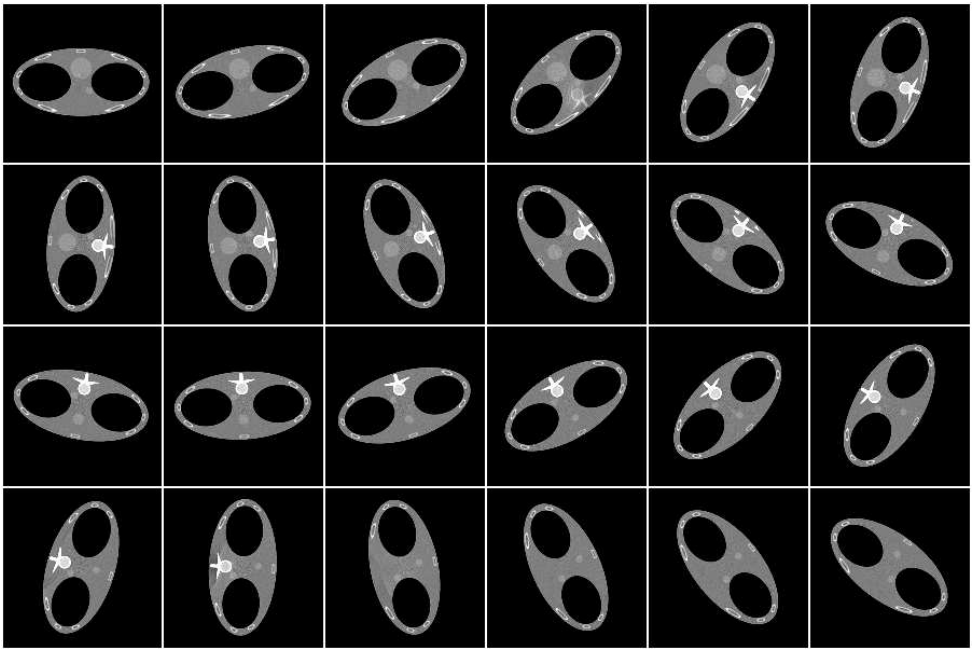


Figure 2. Slices of the thorax phantom before slice rotation.

Listing 3: Fast algorithm with slice increments.

```
void SpiralBP( int const I, // Number of x-pixels x
               int const J, // Number of y-pixels y
               int const K, // Number of slices   z
               int const L, // Number of rows     v
               int const M, // Number of channels u
               int const N, // Number of views    a
               int const dN, // Slice increment
               int const * const mLut, // u(x, y, a)
               int const * const lLut, // v(x, y, a)
             float const * const wLut, // w(x, y, a)
             float const * const Raw,  // p(alpha+a, u, v)
             float       * const Vol)  // f(x, y, alpha)
  {
  #define mlut(i, j, n) mLut[(i*J+j)*N+n]
  #define llut(i, j, n) lLut[(i*J+j)*N+n]
  #define wlut(i, j, n) wLut[(i*J+j)*N+n]
  #define V(i, j, k)     Vol[(i*J+j)*K+k]
  #define R(l, m, dn, kk, k) Raw[((dn*L+l)*M+m)*KTot+kk+k]

  int const KK=N/dN;
  int const KTot=KK+K;

  for(int i=0; i<I; i++) // x-loop
  for(int j=0; j<J; j++) // y-loop
  for(int dn=0; dn<dN; dn++) // a-loop (dn-part)
  for(int kk=0; kk<KK; kk++) // a-loop (kk-part)
       {
       int const n=kk*dN+dn; // compose n using kk and dn

          int const m=mlut(i, j, n); // u(x, y, a)
```

```
        int const l=llut(i, j, n); // v(x, y, a)
      float const w=wlut(i, j, n); // w(x, y, a)

      for(int k=0; k<K; k++) // z-loop
          V(i, j, k)+=w*R(l, m, dn, kk, k); // update voxel value
      }
#undef mlut
#undef llut
#undef wlut
#undef V
#undef R
}
```

4. Results

For our studies, a third-generation clinical CT scanner geometry with $R_F = 600$ mm, $R_D = 450$ mm and $R_M = 400$ mm was used. During one rotation, $N_{360} = 512$ projections were simulated with a table feed of $d = 32$ mm per rotation. This corresponds to a spiral pitch value of 1. We use a detector with $M = 512$ channels of 0.9 mm width and $L = 64$ detector rows with a slice thickness of 0.5 mm (dimensions corresponding to the isocenter).

The patient data were acquired with a Siemens Sensation 64 scanner (Siemens Healthcare, Forchheim, Germany). This scanner has a similar geometry as used in our simulations but it acquires $N_{360} = 1160$ projections per rotation with each detector row consisting of $M = 672$ channels.

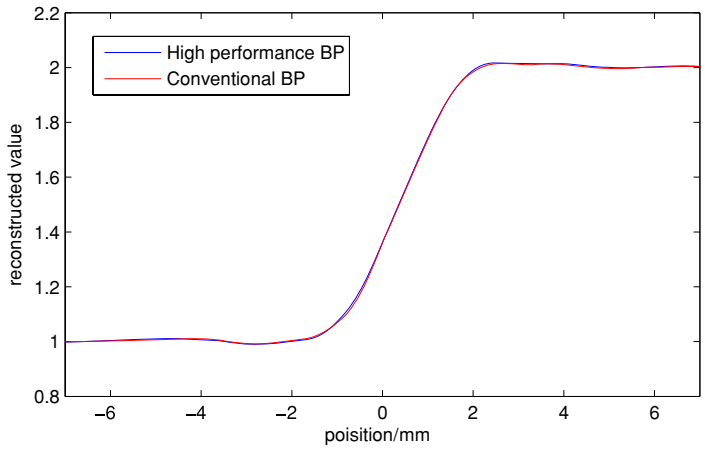
Images were reconstructed using the EPBP reconstruction algorithm, which is an approximate Feldkamp-type image reconstruction algorithm for spiral (and sequential) CT with a flexible voxel-specific weighting scheme that can allow for either 100% dose usage or phase-correlated imaging or it can be used to only backproject data from the minimal data window (Pi window) (Kachelrieß *et al* 2004). It should be noted that any other existing spiral image reconstruction algorithm can be used together with our high performance backprojection as well. The reconstructed volume consists of 512^3 voxels.

4.1. Image quality

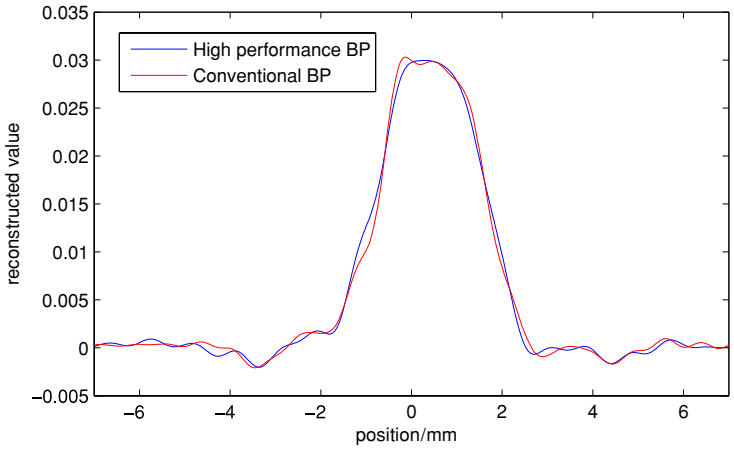
Since our high performance backprojection approach requires an additional slice rotation step, we have to analyze whether this additional step changes the image quality (spatial resolution and noise). In our examples, we use a simple destination-driven rotation algorithm with bilinear interpolation on the source side. We compensate for the frequency characteristics of this bilinear interpolation by enhancing high frequencies in the image reconstruction kernel that is applied during convolution prior to backprojection. The reader interested in rotation algorithms in particular and in resampling and interpolation algorithms in general may refer to one of Maeland (1988), Di Bella *et al* (1996) and Lehmann *et al* (1999).

To compare image quality, we define the reference gold-standard image to be an image reconstructed with a conventional backprojector, i.e. a backprojector that does not require the slice rotation step. The reference image represents the maximum image quality achievable by the selected reconstruction algorithm.

A cylindrical phantom filled with water was simulated and used to evaluate image noise. Table 2 shows that both the conventional backprojection and our high performance approach yield identical image noise.



(a) Edge spread function $p(x)$



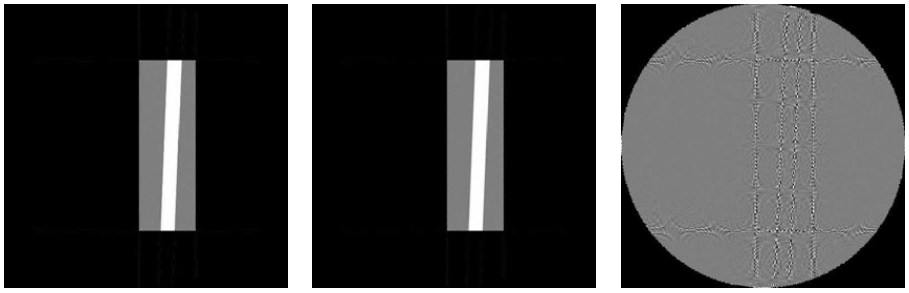
(b) Line spread function $p'(x)$

Figure 3. The edge spread function and the line spread function of a reconstruction driven by our high performance backprojection compared to a reconstruction with a conventional backprojection that does not require the slice rotation step. (a) Edge spread function $p(x)$ and (b) line spread function $p'(x)$.

Table 2. The evaluation of image quality shows that both reconstructions are identical. Due to the modified kernel, the slice rotation does neither increase nor decrease noise or spatial resolution.

Backprojection type	Image noise	Spatial resolution
Conventional	58.0 ± 1.0 HU	0.947 ± 0.02 mm
High performance	57.0 ± 1.0 HU	0.945 ± 0.02 mm

The spatial resolution was assessed using reconstructions of a slanted edge. Figure 4 shows a reconstructed image of our phantom. The edge is slanted to avoid aliasing in this source-driven resampling process, which extracts the edge profile $p(x)$, and to be independent of the pixel size chosen for reconstruction. The profile is finally derived with respect to x to obtain the line spread function.



(a) (b) (c)

Figure 4. The resolution test phantom (with −1000 HU for air, 0 HU for the phantom body and 1000 HU for the slanted edge part). The images are displayed with $C = 0$ HU and $W = 1000$ HU and the difference image uses $C = 0$ HU and $W = 250$ HU. (a) Conventional backprojection reconstruction, (b) high performance backprojection reconstruction and (c) difference.

We determine the moments of $p'(x)$ as

$$\langle x \rangle = \int \mathrm{d}x\, x p'(x) \qquad \langle x^2 \rangle = \int \mathrm{d}x\, x^2 p'(x)$$

and use $\sigma^2 = \langle x^2 \rangle - \langle x \rangle^2$ as a measure of spatial resolution.

Table 2 shows that the high performance backprojection with its final rotation step does not introduce any loss in spatial resolution. This result is confirmed regarding figure 3 which shows the edge spread function and the line spread function for both approaches, a conventional backprojector (without slice rotation) and the high performance spiral backprojector (with slice rotation).

If we have a look at the difference images in figure 4, we recognize some aliasing noise located at the edges of the objects. Since this is natural behavior (aliasing occurs in both reconstructions) and since we cannot appreciate these subtle aliasing artifacts in the original images but only in difference images, we consider the influence of the slice rotation to be negligible.

The semi-anthropomorphic FORBILD thorax phantom[1] was simulated and reconstructed to demonstrate the image quality in a more realistic setting. The images and the difference image in figure 5 do not show any conspicuousness or additional artifacts.

To demonstrate the quality of the algorithm in the clinical routine, a dataset from a Siemens Sensation 64 scanner was reconstructed with the reference and with the high performance algorithm (figure 6). In contrast to the two simulated phantoms, the clinical data contain noise. The difference image shows no anatomical structures inside the patient; hence, both algorithms provide identical image resolution.

4.2. Time measurements and implementation details

For our time measurements, we used an Intel Xeon 7300 platform with four X7460 hexa core processors running at 2.66 GHz with 3 × 3 MB L2-Cache and 16 MB L3-Cache and a Celsius R650 workstation (Fujitsu Siemens Computers) using the Intel Xeon 5400 platform with two X5460 quad core processors running at 3.16 GHz with 2 × 6 MB Cache. For all measurements, the operating system Windows XP Professional x64 Edition, Microsoft Corporation, USA, was used. We also provide performance values for the CBE with 2 Cell processors running

[1] http://www.imp.uni-erlangen.de/phantoms.

(a) (b) (c)

Figure 5. FORBILD thorax phantom. Gray level $C = 0$ HU, $W = 500$ HU for the images and $C = 0$ HU, $W = 100$ HU for the difference image. (a) Conventional backprojection reconstruction, (b) high performance backprojection reconstruction and (c) difference.

(a) (b) (c)

Figure 6. Patient data. The images are scaled with $C = 0$ HU, $W = 500$ HU and with $C = 0$ HU, $W = 100$ HU for the difference image. (a) Conventional backprojection reconstruction, (b) high performance backprojection reconstruction and (c) difference.

at 3.2 GHz with 2 GB RAM provided by Mercury Computer Systems, Chelmsford, MA (operating system Fedora Core 7). The results with the GPU were measured with an NVIDIA Quadro FX 5600, NVIDIA Corporation, Santa Clara, CA, with a Windows XP Professional x64 Edition running on the host computer.

For a fair comparison between different algorithms, we calculate the total number of voxel updates and divide this number by 1024^3 to get the number of giga updates (GU) the algorithm performs. As indicated in the listings, an update is a summation of the rawdata value to the final voxel. After a time measurement, we calculate the number of giga updates that our algorithms can process per second (giga updates per second (GUPS)); this useful measure was initially proposed in Goddard *et al* (2007). To give a simple and intuitive example of these figures, let us make an example for a simple circle scan: assume that 512 projections per 360° have to be backprojected and let the volume contain 512^3 voxels. This amounts to a total of 64 GU. In the case of a spiral scan the geometric relations are very complex, illumination interruption may occur and therefore no analytical formula can be given to calculate the GU value. We therefore simply use one test run to count the number of updates required for our geometry: 47.8 GU are needed to compute a 512^3 volume when using the EPBP algorithm for reconstruction.

The code measured here is an optimized version which uses well-known techniques such as loop unrolling and subset building to achieve high cache utilization and so best performance. A summary of such optimization tricks can be found in Knaup and Kachelrieß (2007). The algorithm was designed to fit the need of vector calculations, so the code makes use of SSE

Table 3. Timing results for the Xeon 7300 platform. The conventional backprojection algorithm is compared to the high performance backprojection which improves performance by about a factor of 5.

	BP time	Operation count	Performance	Frame rate
Conventional EPBP	25.7 s	48.2 GU	1.87 GUPS	19.7 s^{-1}
High performance EPBP	2.98 s	48.2 GU	16.2 GUPS	172 s^{-1}
High performance 1-Pi	1.49 s	25.3 GU	17.0 GUPS	344 s^{-1}

Table 4. Timing results for the high performance EPBP steps running on the Xeon 7300 platform using 24 threads.

Task	Time
Convolution	0.085 s
Backprojection	2.980 s
Slice rotation	0.054 s

Table 5. Table entries and memory usage needed for reconstructing $K = 256$ slices.

	Entries	Datatype	Memory usage
Weighting table (w)	209×10^6	`float`	800 MB
Look-up tables (u, v)	209×10^6	$2 \times$ `short`	800 MB
Rawdata	116×10^6	`float`	444 MB
Volume data	67×10^6	`float`	256 MB
Sum			2300 MB

intrinsics. The Intel 64 technology enables the usage of more than 4 GByte RAM for data storage and some nice features like more registers for data processing (16 instead of 8 SSE registers in a 32 bit mode). The code was compiled using a 64 bit compiler (Microsoft Visual Studio 2005, Microsoft Corporation, USA). Compiling and running under a 32 bit system yield a performance penalty of about 20%. For multiprocessing (we have up to 24 cores for data processing), the code is parallelized with OpenMP directives. The implementation divides the volume into subvolumes, and these subvolumes were delegated to the cores for processing. Timing measurements and our experience with various memory layouts show that this is the most efficient way with the lowest overhead for backprojection to achieve a good speedup factor.

Results for different algorithm types (using the Xeon 7300 platform) are shown in table 3. A comparison of the highly optimized conventional backprojection (no slice rotation) with the highly optimized high performance backprojection (with rotated slices) shows a nearly ten-fold performance increase, thanks to using the spiral symmetry. The table also compares the backprojection within the EPBP algorithm, which makes use of all rawdata, with the backprojection in a 1-Pi exact image reconstruction algorithm, where only data within the Tam window are used: the performance is nearly identical (16.2 GUPS versus 17.0 GUPS) but due to the low dose efficiency of the exact reconstruction algorithm, less rays have to be backprojected and therefore more images can be provided per second (172 versus 344).

A more detailed timing measurement of the complete high performance EPBP reconstruction given in table 4 shows that the final slice rotation step as well as the convolution is negligible compared to the backprojection.

To give an overview of the resources required, the size of the weighting and look-up tables and of the data arrays is summarized in table 5 for the EPBP case. The number of entries of the

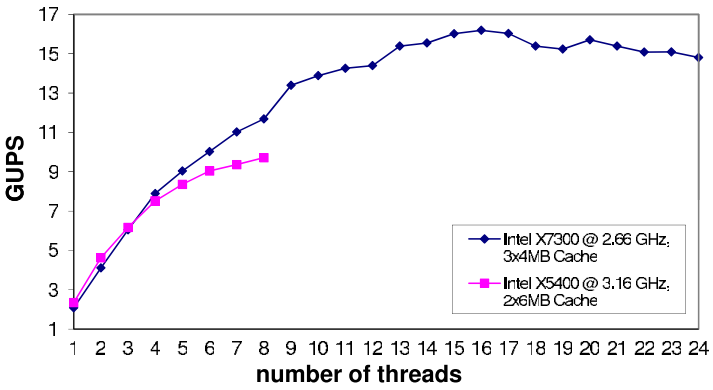


Figure 7. Scaling on the system using multiple threads.

Table 6. Comparison between our different high performance backprojection algorithms and platforms.

Architecture	Cores	Spiral BP	Parallel BP	Perspective BP
Intel Xeon 5400	8	9.7 GUPS	37 GUPS	7.9 GUPS
Intel Xeon 7300	24	16 GUPS	78 GUPS	16 GUPS
Cell (Mercury CBE)	8	23 GUPS	21 GUPS	4.7 GUPS
Two socket Cell (Mercury CBE)	16	46 GUPS	42 GUPS	9.4 GUPS
NVIDIA GeForce 8800 GTX	128	–	14 GUPS	12 GUPS

look-up tables was calculated as $\mathrm{d}N \times I \times J \times (KK + K)$. The parameter KK, representing the numbers of projections contributing to a single slice (see listing 3), is highly dependent on the scan and scanner geometry and on the reconstruction algorithm. KK is determined by test runs of our reconstruction algorithms. For example, we find $KK = 632$ for the EPBP algorithm whereas the 1-Pi window reconstruction yields $KK = 128$. Note that table 5 was generated using $K = 256$. Consequently, it took us two calls to the backprojection function to reconstruct the complete volume with 512 slices.

Our high performance spiral backprojection requires a large memory bandwidth which increases with increased backprojection speed. The amount of calculation required during backprojection is very low compared to standard backprojectors. This is because the high performance spiral backprojector achieves a high data-level parallelism, which means that long vectors are processed at once. To analyze the situation for our architectures, we conducted reconstructions with a varying number of cores or threads (figure 7). Both graphs should increase linearly, if the scalability was perfect. The flattening of the curves with a very large number of threads that share the same memory buses shows that the memory bandwidth, which is limited to about 5 GB s^{-1} for the architectures used, becomes the bottleneck.

As a result of this, other architectures such as vector systems with a higher memory bandwidth are of high interest for this type of algorithm. One example is the CBE that has a memory bandwidth of 25 GB s^{-1}. These processors have eight worker cores (called SPUs) and they are optimized for stream processing with a highly specialized vector instruction set. They can perform a complete vector multiply–add in only one cycle. With the method presented here, these units can work very efficiently and CBE measurements show a very high performance (table 6).

We further observed that our high performance spiral backprojector behaves very similar to our high performance parallel backprojector, which is described in Kachelrieß *et al* (2008),

in those cases where memory bandwidth is not the bottleneck (where only a few threads run in parallel). Both algorithms use complete rods of voxels parallel to the z-axes to vectorize the backprojection. This observation allows for a comparison to GPU implementations of parallel backprojection (we currently have no GPU implementation of the spiral backprojector available). Once the data are on the GPU, the memory bandwidth is rather high. However, table 6 indicates that all architectures outperform the GPU regarding the parallel backprojection, and it is likely that this observation also applies to the spiral backprojection. This shows that the GPU cannot efficiently handle the high level of data parallelism as it occurs in parallel backprojection and in spiral backprojection, if our trick of using the spiral symmetry is used. In cases where the data access pattern is less regular and no data-level parallelism is available, the GPU performs similar or even better than the other architectures. One very prominent example is the perspective cone-beam backprojection. Results for our implementations are given in the last column of table 6.

5. Discussion

The high performance spiral backprojection approach is a general-purpose spiral backprojector since it can be used with any spiral image reconstruction algorithm. It benefits from using the symmetries of the trajectory. The concept itself may be adapted to other trajectories of high symmetry as well.

A drawback of the proposed algorithm is the requirement for circular FOVs that are centered about the rotation center. This drawback is somewhat compensated by the high reconstruction speed of the algorithm. Further, the final slice rotation step may be regarded as a disadvantage because it may impair image quality if not implemented with care. Our experience, however, shows that the slice rotation can be handled very well and allows for the same image quality as one achieves with a standard backprojector without slice rotation. In combination with more sophisticated reconstruction grids, such as the hexagonal grid, for example, the slice is an ideal setting since a resampling to the Cartesian grid has to be done anyway. Computationally, the final slice rotation is costless.

The performance values shown in this paper are only an example valid for this specific scanner geometry and reconstruction algorithm that we used. Our experiments have shown that slight changes of the geometry (e.g. a different cone angle or a different number of slices) may significantly change those values. The numbers cited here are the most conservative ones we observed. The performance increase of about a factor of 10 compared to a conventional backprojection, however, is characteristic for all geometries.

There are several possibilities of reducing the bandwidth requirements. The most simple is to use symmetries of the volume and therefore to reduce the amount of storage required for the weight tables and the look-up tables of the detector coordinates. For example with a Cartesian volume using symmetries can reduce the memory required to store these tables by a factor of 8. With a hexagonal grid, one can even come down to a factor of 12. Further on one may use 16 bit floats, so-called halfs, to store the tables (our current implementation uses 32 bit floats). This would yield another factor of 2. If there is no need for the feature of voxel-specific weighting, one can modify the algorithm to work without the weighting tables. For this case, we found that the reduced memory bandwidth requirements can yield about another 20% performance improvement. The most significant impact on the bandwidth bottleneck is the access to the rawdata, an access pattern that is hardly predictable. The rawdata storage pattern can be improved if one desires to optimize the algorithm for a specific scanner and scan geometry.

Spiral backprojection may further significantly benefit from new compute architectures such as the Intel Larrabee (LRB) architecture (Seiler *et al* 2008). It comprises 16 vectors, instead of the 4 vectors used in our implementation, and the possibility of loading from and storing to 16 bit floats without additional clock cycles. The first feature will increase the performance per clock cycle by a factor of 4 since 16 instead of 4 voxels can be backprojected simultaneously. The second feature will reduce the bandwidth requirements by 50%. Sophisticated scatter/gather capabilities as well as new vector broadcast instructions may further reduce the number of clock cycles per voxel since the loading of weight and look-up values can be significantly improved. Therefore, we expect a significant performance increase using the LRB architecture in the near future.

Our fast spiral backprojection algorithm can be easily redesigned to carry out a forward projection instead of a backprojection. One needs to simply swap the rawdata and the volume access in the innermost loop. Combining such a fast spiral forward projector with our fast spiral backprojector allows for fast spiral iterative image reconstruction. Our current implementations realize a pixel-driven backprojection as well as the listings shown in this paper. It is also easy to redesign the backprojector or the forward projector to realize a ray-driven strategy.

5.1. Other publications

Recently, other implementations for spiral CT were published (comparisons to parallel-beam or perspective cone-beam backprojection algorithms can be found in Kachelrieß *et al* (2007)).

One group uses an HP 6200 workstation with 2 Intel Xeon 3.2 GHz processors for their backprojection; the geometry is the standard Siemens geometry with 1160 projections and 672 detector channels (Zeng *et al* 2007). They backproject one slice with 512×512 pixels with their optimized algorithm within 1.32 s. This reconstruction has an operational count of 0.28 GU and given the speed of 1.32 s per slice, we find that they achieve a performance of 0.21 GUPS. Three main speedup tricks to gain a higher performance were used: the use of 90° symmetries, the optimization of the data structures using SIMD instructions and multithreading.

A GPU-based spiral image reconstruction was analyzed in Bi *et al* (2008). The authors report about reconstructions of simulated data (256 slices) and measured data (216 slices) with a pitch value of 1. Each slice has 256×256 pixels and there are 360 projections (simulation) and 720 projections (measurement) per rotation to backproject. With a voxel update number of 5.6 GU for the simulation and 9.5 GU for the measurement, this is a rather small problem. They run their code on two different GPUs; the best results are obtained with an NVIDIA GeForce 8800 GTX where they observe up to 1.5 GUPS. Their CPU-based reference code achieves only 0.07 GUPS.

6. Summary and conclusion

The performance of spiral cone-beam image reconstruction is improved using our high performance general-purpose spiral backprojection approach. Due to using the spiral symmetry in an elegant way, we achieve a high level of vectorization combined with parallelization. The approach is compatible with today's CPUs; it performs very well on the CBE, probably slightly less well on the GPU, and it appears to be highly suited for the LRB architecture. With simple redesigns of the spiral backprojector, one can generate fast algorithms for spiral forward projection and one can also realize a ray-driven processing.

A nearly ten-fold performance improvement was observed when comparing this new approach to our highly optimized conventional spiral cone-beam backprojector. Compared to other publications, our high performance approach seems to be faster by order(s) of magnitude. The image quality achievable with the new backprojector is identical to conventional reconstructions.

Acknowledgments

This work was supported by Visage Imaging GmbH, Berlin, Germany. Parts of the reconstruction software were provided by RayConStruct GmbH, Nürnberg, Germany. We thank the Intel Corporation for providing their latest Xeon 7300 multi-core platforms and for providing early access to Larrabee-related information. We thank Fujitsu Siemens Computers GmbH, Germany, for providing their Celsius R650 Workstation which is based on the Xeon 5400 platform.

References

Bi W, Chen Z, Zhant L and Xing Y 2008 Accelerate helical cone-beam CT with graphics hardware *Proc. SPIE* **6913** (published online)

Bontus C, Köhler T and Proksa R 2003 A quasiexact reconstruction algorithm for helical CT using a 3Pi acquisition *Med. Phys.* **30** 2493–502

Danielsson P-E, Edholm P, Eriksson J, Magnusson-Seger M and Turbell H 1998 Helical cone beam scanning an reconstruction of long objects using 180° exposure *Patent Application* PCT/SE 98/00029

Defrise M, Noo F and Kudo H 2000 A solution to the long-object problem in helical cone-beam tomography *Phys. Med. Biol.* **45** 623–43

Di Bella E V R, Barclay A B, Eisner R L and Schafer R W 1996 A comparison of rotation-based methods for iterative reconstruction algorithms *IEEE Trans. Nucl. Sci.* **43** 3370–6

Goddard I, Berman A, Bockenbach O, Lauginiger F, Schuberth S and Thieret S 2007 Evolution of computer technology for fast cone beam backprojection *Proc. Computational Imaging Conf.*

Kachelrieß M, Knaup M and Bockenbach O 2007 Hyperfast parallel-beam and cone-beam backprojection using the cell general purpose hardware *Med. Phys.* **34** 1474–86

Kachelrieß M, Knaup M and Kalender W A 2004 Extended parallel backprojection for standard 3D and phase-correlated 4D axial and spiral cone-beam CT with arbitrary pitch and 100% dose usage *Med. Phys.* **31** 1623–41

Kachelrieß M, Knaup M, Marone F and Stampanoni M 2008 Hyperfast O(2048^4) image reconstruction for synchrotron-based x-ray tomographic microscopy *IEEE Med. Imaging Conf. Rec.* **M06-85** 3810–3

Kalender W 2005 *Computed Tomography* 2nd edn (New York: Wiley)

Katsevich A 2002 Theoretically exact FBP-type inversion algorithm for spiral CT *SIAM J. Appl. Math.* **62** 2012–26

Katsevich A 2006 3Pi algorithms for helical computed tomography *Adv. Appl. Math.* **36** 213–50

Knaup M and Kachelrieß M 2007 Acceleration techniques for 2D parallel and 3D perspective forward and backprojections *9th Int. Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine* pp 45–8

Knaup M, Steckmann S, Bockenbach O and Kachelrieß M 2007 CT image reconstruction using hexagonal grids *IEEE Med. Imaging Conf. Rec.* **M13-277** 2074–3076

Kudo H, Noo F, Defrise M and Rodet T 2003 New approximate filtered backprojection algorithm for helical cone-beam CT with redundant data *IEEE Nucl. Sci. Symp. Conf. Rec.* **5** 3211–5

Lehmann T M, Gönner C and Spitzer K 1999 Survey: interpolation methods in medical image processing *IEEE Trans. Med. Imaging* **18** 1049–75

Maeland E 1988 On the comparison of interpolation methods *IEEE Trans. Med. Imaging* **7** 213–7

Noo F, Pack J and Heuscher D 2003 Exact helical reconstruction using native cone-beam geometries *Phys. Med. Biol.* **48** 3787–818

Proksa R, Köhler T, Grass M and Timmer J 2000 The n-pi-method for helical cone-beam CT *IEEE Trans. Med. Imaging* **19** 848–63

Seiler L *et al* 2008 Larrabee: a many-core x86 architecture for visual computing *ACM Trans. Graph.* **27** 18:1–18:15

Steckmann S, Knaup M and Kachelrieß M 2008a Hyperfast general-purpose cone-beam spiral backprojection with voxel-specific weighting *IEEE Med. Imaging Conf. Rec.* **M11-3** 5426–33

Steckmann S, Knaup M and Kachelrieß M 2008b Algorithm for hyperfast cone-beam spiral backprojection *11th Int. Conf. on Medical Image Computing and Computer Assisted Intervention, Workshop High-Performance Medical Image Computing and Computer Aided Intervention*

Taguchi K, Chiang B S and Silver M D 2004 A new weighting scheme for cone-beam helical CT to reduce the image noise *Phys. Med. Biol.* **49** 2351–64

Tang X, Hsieh J, Nilsen R A, Dutta S, Samsonov D and Hagiwara A 2006 A three-dimensional-weighted cone beam filtered backprojection (CB-fBP) algorithm for image reconstruction in volumetric CT-helical scanning *Phys. Med. Biol.* **51** 855–74

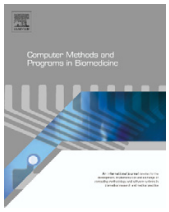
Turbell H 2001 Cone-beam reconstruction using filtered backprojection *PhD Thesis* Linköping Universitet

Zeng K, Bai E and Wang G 2007 A fast CT reconstruction scheme for a general multi-core PC *J. Biomed. Imaging* **2007** 1–9

COMM-2960; No. of Pages 8 ARTICLE IN PRESS

COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE XXX (2009) XXX–XXX

journal homepage: www.intl.elsevierhealth.com/journals/cmpb



Algorithm for hyperfast cone-beam spiral backprojection

Sven Steckmann, Michael Knaup, Marc Kachelrieß*

Institute of Medical Physics (IMP), University of Erlangen-Nürnberg, Henkestr. 91, 91052 Erlangen, Germany

ARTICLE INFO

Article history:
Received 3 February 2009
Received in revised form
14 July 2009
Accepted 14 August 2009

Keywords:
CT
Image reconstruction
Backprojection
Spiral CT
High performance computing

ABSTRACT

Cone-beam spiral backprojection is computationally highly demanding. At first sight, the backprojection requirements are similar to those of cone-beam backprojection from circular scans such as it is performed in the widely used Feldkamp algorithm. However, there is an additional complication: the illumination of each voxel, i.e. the range of angles the voxel is seen by the X-ray cone is a complex function of the voxel position. The weight function has no analytically closed form and must be numerically determined. Storage of the weights is prohibitive since the amount of memory required equals the number of voxels per spiral rotation times the number of projections a voxel receives contributions and therefore is in the order of 10^9 to 10^{11} floating point values for typical spiral scans. We propose a new algorithm that combines the spiral symmetry with the ability of today's 64 bit CPUs to store large amounts of precomputed weights. Using the spiral symmetry in this way allows to exploit data-level parallelism and thereby to achieve a very high level of vectorization. An additional postprocessing step rotates these slices back to normal images. Our new backprojection algorithm achieves up to 24.6 Giga voxel updates per second (GUPS) on our systems that are equipped with two standard Intel X5570 quad core CPUs (Intel Xeon 5500 platform, 2.93 GHz, Intel Corporation). This equals the reconstruction of 410 images per second assuming each slice consists of 512×512 pixels, receiving contributions from 512 projections.

1. Introduction

High performance image reconstruction (HPIR) basically means high performance backprojection since backprojection is the most demanding step in the image reconstruction pipeline. Recently, we published our work on high performance parallel beam backprojection and high performance perspective cone-beam backprojection using cell broadband engine (CBE) based implementations and central processing unit (CPU) based implementations [1].

Spiral backprojection is, however, more complicated since the illumination of the voxels by the cone exhibits a complex voxel location-dependent behavior that must be taken into account during the backprojection step. Basically, this requires to compute a weight function $w(x, y, z, \alpha)$, where (x, y, z) is the voxel position and α is the angle of the ray that is backprojected, and apply this function during the backprojection. Numerous algorithms have been published that already use voxel-specific weighting [2–5]. Others assume some simplifications to circumvent the problem of voxel-specific weighting and thereby compromise either dose usage or image quality [6–12]. Also implementations of quasiexact and exact reconstruction algorithms such as [13–15] may profit from our new approach.

The paper proposes the new algorithm [16] and extends the paper [17] by implementation details for a filtered backprojection. The new backprojection algorithm is able to perform the correct voxel weighting and that can be used together with

* *Corresponding author*. Tel.: +49 91318525830.
E-mail address: sven.steckmann@imp.uni-erlangen.de (S. Steckmann).
0169-2607/$ – see front matter © 2009 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.cmpb.2009.08.003

COMM-2960; No. of Pages 8
ARTICLE IN PRESS
2 COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE XXX (2009) XXX–XXX

any spiral reconstruction algorithm, be it exact, approximate or iterative (in which case a corresponding forward projector must be programmed as well). This paper does not propose a new type of image reconstruction.

2. Backprojection

Let α be the projection angle. The source position as a function of α is given as

$$\mathbf{s}(\alpha) = \begin{pmatrix} +R_F \sin\alpha \\ -R_F \cos\alpha \\ đ\alpha \end{pmatrix} \tag{1}$$

where $đ = d/2\pi$ and d is the table increment per full rotation and R_F the radius of the focal trajectory. Backprojection means evaluating

$$f(x,y,z) = \int d\alpha w(x,y,z,\alpha)\hat{p}(\alpha,u,v) \tag{2}$$

with $u = u(x,y,z,\alpha)$ and $v = v(x,y,z,\alpha)$ being the detector's lateral and longitudinal coordinates given by the intersection of the ray from $\mathbf{s}(\alpha)$ through the voxel (x,y,z) with the detector surface. $\hat{p}(\alpha,u,v)$ is the convolved projection data. For an overview see also Fig. 1.

Our aim is to precompute and store the weight function $w(x,y,z,\alpha)$ as well as the detector look-up coordinates $u(x,y,z,\alpha)$ and $v(x,y,z,\alpha)$ before image reconstruction. Our aim further is to vectorize the backprojection algorithm.

To achieve full vectorization of the backprojection algorithm and to reduce the memory required to store the voxel-specific weights we now use the fact that the system has a spiral symmetry. Let

$$\hat{f}(x,y,\alpha) = f(x\cos\alpha - y\sin\alpha,\ x\sin\alpha + y\cos\alpha,\ đ\alpha) \tag{3}$$

be a new representation of the volume f. In the new volume $\hat{f}$ each slice rotates in the same fashion as the spiral trajectory. Hence $\hat{f}$ exhibits the same spiral symmetry as the data acquisition itself. The slice's z-position is determined by the projection angle α. Now, backprojection is

$$\hat{f}(x,y,\alpha) = \int da w(x,y,a)\hat{p}(\alpha+a,u,v) \tag{4}$$

with $u = u(x,y,a)$ and $v = v(x,y,a)$. The angle a is counting relative to the slice position.

This new backprojection equation is much more favorable to implementation since neither the integrand's weight function nor the look-up values u and v depend on the absolute angle α, they only depend on the relative angle a. Hence we can loop over a range of z-positions α and add the corresponding projection entries to the volume without reevaluation of the weighting or look-up values. If we represent $\hat{f}$ and $\hat{p}$ with α being the linear, and thus fast, variable in memory we achieve a fully vectorization of the backprojection. The spiral symmetry further helps to reduce the weight table w and the look-up positions u and v from four-dimensional arrays to significantly smaller three-dimensional tables. These can be easily held in memory of modern PCs where 16–32 GB of memory are typically available. The (non-optimized) reference source code of Listing 1 illustrates the algorithm for backprojection.

The listing clearly shows why this algorithm is good for vectorization. The innermost loop, i.e. the loop over `k`, is the fastest index. For treating this loop as a vector, the data belonging to this loop must be arranged linear in memory. This can be seen from the data layout as specified by `#define` directives: the index `k` is a simple offset in the data and thus accesses are linear in memory.

The limitation of this version of the algorithm is that the distance of adjacent slices is fixed to the increment Δz of the scanner between two adjacent projections. This problem can be resolved by a new variable `dN` that counts how many multiples of Δz the slices shall be separated. The extend algorithm

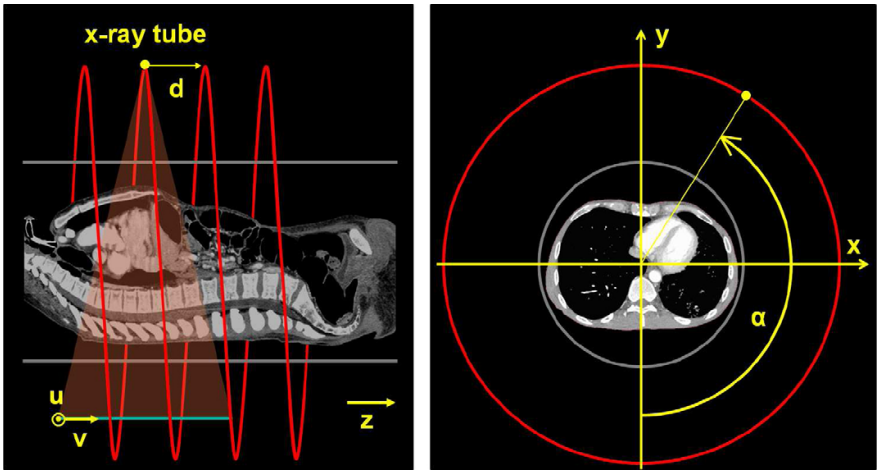


Fig. 1 – Illustration of a Spiral-CT scan with the table feed per rotation of *d* and *u* and *v* as the detector. The position of the tube is here on the top, the detector on the bottom.

COMM-2960; No. of Pages 8 ARTICLE IN PRESS
COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE XXX (2009) XXX–XXX

is given in Listing 2. Note that adding the freedom to specify the longitudinal sampling requires only the simple decomposition of the view number `n` into `n=kk*dN+dn`. This is nothing but a reordering of our projections.

Note that the innermost loop remains to be fast and vectorizeable. The main difference to our Listing 1 is that the memory layout of the rawdata has become slightly more complex and is now five-dimensional.

Listing 1: Reference backprojection algorithm.

```
void SpiralBP(  int const I, // Number of x-pixels x
                int const J, // Number of y-pixels y
                int const K, // Number of slices   z
                int const L, // Number of rows     v
                int const M, // Number of channels u
                int const N, // Number of views    a
                int const * const mLut, // u(x, y, a)
                int const * const lLut, // v(x, y, a)
              float const * const wLut, // w(x, y, a)
              float const * const Raw,  // p(alpha+a, u, v)
              float       * const Vol)  // f(x, y, alpha)
    {
    #define mlut(i, j, n) mLut[(i*J+j)*N+n]
    #define llut(i, j, n) lLut[(i*J+j)*N+n]
    #define wlut(i, j, n) wLut[(i*J+j)*N+n]
    #define V(i, j, k)     Vol[(i*J+j)*K+k]
    #define R(l, m, n, k) Raw[((l*M+m)*(N+K)+n+k]

    for(int i=0; i<I; i++) // x-loop
    for(int j=0; j<J; j++) // y-loop
    for(int n=0; n<N; n++) // alpha-loop
        {
          int const m=mlut(i, j, n); // u(x, y, a)
          int const l=llut(i, j, n); // v(x, y, a)
        float const w=wlut(i, j, n); // w(x, y, a)

        for(int k=0; k<K; k++)
            V(i, j, k)+=w*R(l, m, n, k); // z-loop
        }

    #undef mlut
    #undef llut
    #undef wlut
    #undef V
    #undef R
    }
```

3. Slice rotation

The improved backprojection performance comes at the price of having the slices rotating together with the spiral. This requires us to add an additional rotation step. A more significant disadvantage of the approach is that the field of view (FOV), i.e. the region of the field of measurement (FOM) that shall be reconstructed, must be centered in the isocenter and it will be circular. To obtain rectangular FOVs some clipping must occur after the rotation step. In comparison to the high reconstruction speed, this does not carry any weight if we drop some reconstructed voxels.

3.1. *Rotation algorithm*

To rotate the slices back we have to implement the transform

$$f(x, y, z) = \hat{f}(x\cos\alpha + y\sin\alpha,\ y\cos\alpha - x\sin\alpha,\ \alpha). \tag{5}$$

The image processing literature suggests many possible solutions to rotate images [18–20]. To be fast and accurate we use a simple destination-driven resampling of the rotated images with a two-by-two point interpolation. This interpolation uses a trapezoidal function to prevent loss of spatial resolution and to avoid creating aliasing. Slice rotation requires interpolation only in the lateral variables and no longitudinal interpolation since the rotated slices are generated just at the required z-positions (i.e. the values of α are chosen to match the required slice positions z).

Listing 2: Algorithm with slice increments.

```
void SpiralBP(  int const I, // Number of x-pixels x
                int const J, // Number of y-pixels y
                int const K, // Number of slices   z
                int const L, // Number of rows     v
                int const M, // Number of channels u
                int const N, // Number of views    a
                int const dN, // Slice increment
                int const * const mLut, // u(x, y, a)
                int const * const lLut, // v(x, y, a)
              float const * const wLut, // w(x, y, a)
              float const * const Raw,  // p(alpha+a, u, v)
              float       * const Vol)  // f(x, y, alpha)
  {
  #define mlut(i, j, n) mLut[(i*J+j)*N+n]
  #define llut(i, j, n) lLut[(i*J+j)*N+n]
  #define wlut(i, j, n) wLut[(i*J+j)*N+n]
  #define V(i, j, k)     Vol[(i*J+j)*K+k]
  #define R(l, m, dn, kk, k) Raw[((dn*L+l)*M+m)*KTot+kk+k]

  int const KK=N/dN;
  int const KTot=KK+K;

  for(int i=0; i<I; i++) // x-loop
  for(int j=0; j<J; j++) // y-loop
  for(int dn=0; dn<dN; dn++) // a-loop (dn-part)
  for(int kk=0; kk<KK; kk++) // a-loop (kk-part)
      {
      int const n=kk*dN+dn; // compose n using kk and dn

        int const m=mlut(i, j, n); // u(x, y, a)
        int const l=llut(i, j, n); // v(x, y, a)
      float const w=wlut(i, j, n); // w(x, y, a)

      for(int k=0; k<K; k++) // z-loop
          V(i, j, k)+=w*R(l, m, dn, kk, k);
      }
  #undef mlut
  #undef llut
  #undef wlut
  #undef V
  #undef R
  }
```

For interpolation we use the trapezoidal function $T_w(\chi)$. It consists of a plateau of size $1-w$ and has a full width of $1+w$. Mathematically,

$$T_w(\chi) = \frac{1}{2w}\begin{cases} 0 & \text{if } |2\chi| > 1+w \\ 1+w-|2\chi| & \text{else if } |2\chi| > 1-w \\ 2w & \text{else} \end{cases} \tag{6}$$

The parameter χ stands for the index domain where interpolation occurs (e.g. for pixel columns i or rows j). Note that $w=0$ results in nearest neighbor interpolation while $w=1$ gives a linear interpolation algorithm.

The slice rotation algorithm uses the parameter $w=0.5$ for interpolation in the x- and y-direction.

3.2. *Kernel modifications*

To compensate for the smoothing caused by the interpolation during slice rotation we have to modify the reconstruction kernel to return to the original spatial resolution and image noise level. An empirical approach is used to do so. To describe the smoothing of the point spread function during the interpolation we used a convolution with a Gaussian function. In

Fourier domain this means:

$$\hat{K}_m = K_m \, e^{a\,m^2}. \tag{7}$$

K_m, with m being the channel index, represents the standard kernel (in Fourier domain) that would be used for a conventional spiral backprojection algorithm without slice rotation. $\hat{K}_m$ is our kernel modification. The parameter $a > 0$ is empirically chosen to match the spatial resolution and the image noise to the noise obtained with a standard backprojection algorithm.

3.3. *More advanced grids*

Since slice rotation requires a final rotation and therefore a final resampling step and since this requires an empirical kernel adaptation one can also think of doing the complete backprojection on a non-Cartesian grid at no additional costs. Resampling to the Cartesian grid can then be done together with the slice rotation at no additional cost. As presented in Ref. [21] a hexagonal grid has several advantages over the standard Cartesian sampling. In comparison to the Cartesian grid, the hexagonal grid needs only $\sqrt{3}/2 \approx 0.866$ times the number of pixels for the same sampling density. To provide a better understanding and traceability for the reader with well known techniques, the results shown here are calculated on a standard cartesian grid.

4. Results

For our studies a 3rd generation scanner geometry with $R_F = 600\,\text{mm}$, $R_D = 450\,\text{mm}$ and $R_M = 800\,\text{mm}$ was used. During one rotation $N_{360} = 512$ projections were simulated with a table feed of $d = 32\,\text{mm}$ per rotation. This corresponds to a pitch of one. We are using a detector with $M = 512$ channels of 0.9 mm width and $L = 64$ detector rows with a slice thickness of 0.5 mm (dimensions corresponding to the isocenter).

The patient data were acquired with a Siemens Sensation 64 scanner (Siemens Healthcare, Forchheim, Germany). This scanner has the same geometry as used in our simulations but it acquires $N_{360} = 1160$ projections per rotation with each detector row consisting of $M = 672$ channels.

Images were reconstructed using the EPBP reconstruction algorithm, which is an approximate Feldkamp-type image reconstruction algorithm for spiral (and sequential) CT with a flexible voxel-specific weighting scheme that can allow for either 100% dose usage or for phase-correlated imaging or it can be used to only backproject data from the minimal data window (Pi window) [2]. The reconstructed volume consists of 512^3 voxels.

Table 1 – The evaluation of image quality shows that both reconstructions are identical. Due to the modified kernel the slice rotation does neither increase nor decrease noise or spatial resolution.

Algorithm type	Image noise	Spatial resolution
Conventional backprojection	58.0 HU	0.947 mm
Hyperfast backprojection	57.0 HU	0.945 mm

4.1. *Image quality*

Since our hyperfast general purpose backprojection approach requires an additional slice rotation step we have to analyze whether this additional step changes the image quality (spatial resolution and noise).

To compare image quality we define the reference gold-standard image to be an unrotated image, i.e. an image before the rotation step that does not need any rotation (apart from multiples of 90° which does not require interpolation). The reference image represents the maximum image quality achievable by the selected reconstruction algorithm.

To measure the noise propagation and compare it to our standard, a simulated cylindrical phantom filled with water was simulated and evaluated. Table 1 shows that the modified reconstruction kernel correctly compensates for differences in image noise due to the rotation step: both the conventional backprojection and our hyperfast approach yield identical image noise.

The spatial resolution was assessed using reconstructions of a slanted edge. Fig. 3 shows such an image. To measure the spatial resolution the top edge of the inner rhomboid was used. Table 1 shows that the hyperfast backprojection with its final rotation step does not introduce any loss in spatial resolution. This result is confirmed regarding Fig. 2 which shows the edge spread function and the pseudo point spread function for both approaches, a conventional backprojector (without

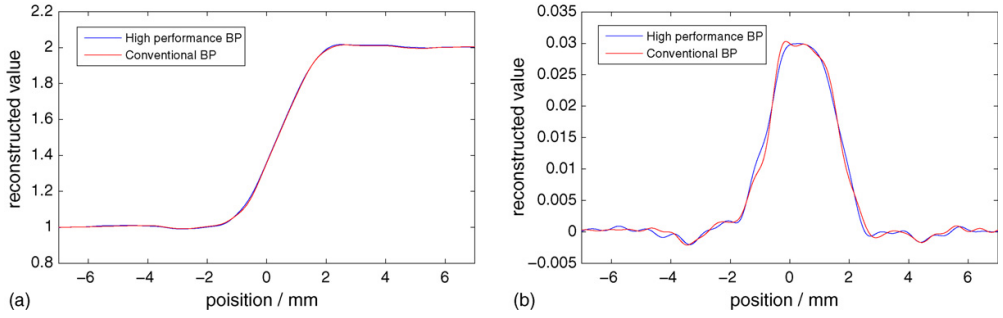


Fig. 2 – The edge spread function and the pseudo point spread function of an example image. (a) Edge spread function $p(x)$ and (b) pseudo point spread function $p'(x)$.

COMM-2960; No. of Pages 8 ARTICLE IN PRESS
COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE XXX (2009) XXX–XXX

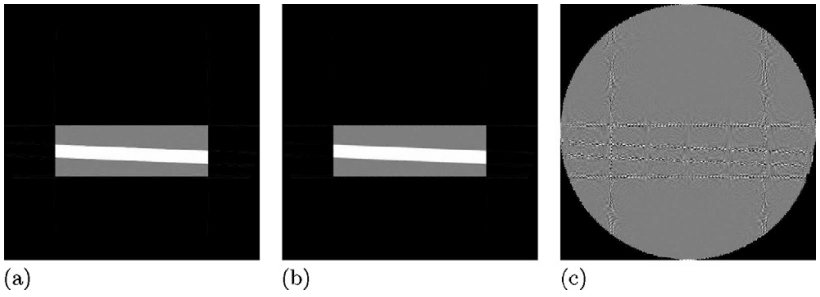


Fig. 3 – The resolution test phantom (with −1000 HU for air, 0 HU for the phantom body and 1000 HU for the slanted edge part). The images are displayed with $C = 0$ HU and $W = 1000$ HU and the difference image uses $C = 0$ HU and $W = 250$ HU. (a) Conventional backproject reconstrucion, (b) hyperfast backproject reconstrucion and (c) reference.

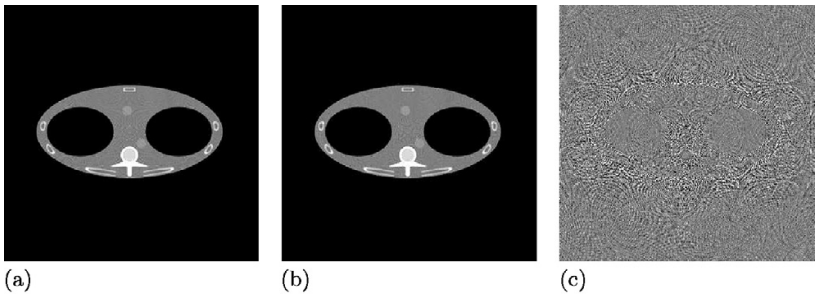


Fig. 4 – FORBILD thorax phantom. Gray level $C = 0$ HU, $W = 500$ HU for the images and $C = 0$ HU, $W = 100$ HU for the difference image. (a) Conventional backprojection reconstruction, (b) hyperfast backprojection reconstruction and (c) difference.

slice rotation) and the hyperfast spiral backprojector with slice rotation. If we have a look at the difference images in Fig. 3 we recognize some aliasing noise located at the edges of the objects. Since this is a natural behavior and since we cannot appreciate these subtle aliasing artifacts in the original images but only in difference images we consider the influence of the slice rotation to be negligible.

The semianthropomorphic FORBILD thorax phantom (http://www.imp.uni-erlangen.de/phantoms) was simulated and reconstructed to demonstrate the image quality in a more realistic setting. The images and the difference image in Fig. 4 does not show any conspicuousness or additional artifacts.

To demonstrate the quality of the algorithm in the clinical routine, a dataset from a Siemens Sensation 64 scanner was reconstructed with the reference and with the hyperfast algorithm (Fig. 5). In contrast to the two simulated phantoms, the clinical data contain noise. The difference image shows no anatomical structures inside the patient: hence both algorithms provide identical image quality.

4.2. *Time measurements and implementation details*

For our time measurements we used an Intel Xeon 5500 platform with two X5570 processors running at 2.93 GHz. For

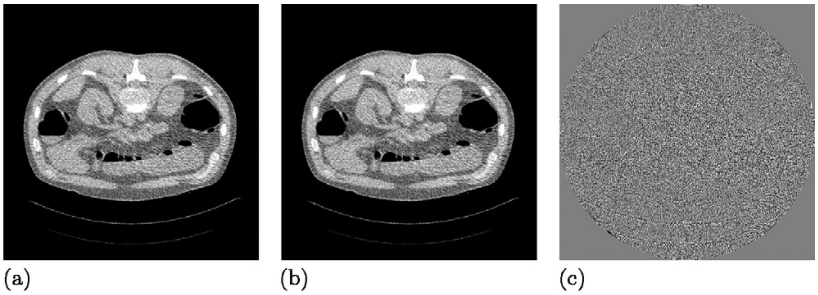


Fig. 5 – Patient data. The images are scaled with $C = 0$ HU, $W = 500$ HU and with $C = 0$ HU, $W = 100$ HU for the difference image. (a) Conventional backprojection reconstruction, (b) hyperfast backprojection reconstruction and (c) difference.

COMM-2960; No. of Pages 8
ARTICLE IN PRESS
6 COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE XXX (2009) XXX–XXX

Table 2 – Timing results. The conventional backprojection algorithm is compared to the hyperfast backprojection which improves performance by about a factor of eleven.

	BP time	Operation count	Performance	Frame rate
EPBP (conv.)	31.3 s	53.1 GU	1.69 GUPS	16.4 s^{-1}
EPBP	2.84 s	53.1 GU	18.7 GUPS	180 s^{-1}
1-Pi	1.25 s	30.7 GU	24.6 GUPS	410 s^{-1}

all measurements the operating system Windows XP Professional x64 Edition, Microsoft Corporation, USA, was used. The results are shown in Table 2. For a fair comparison between different algorithms we calculate the total numbers of updates and divide them by 1024^3 to get our giga updates (GU). An update is a summation of the rawdata value to the final volume. After a time measurement we calculate the giga updates we can handle per second (GUPS, giga updates per second). For a spiral trajectory this is not as easily to calculate as for a circle, so let us make an example for the circle trajectory. First: assume we are using 512 projections per 360°. Our volume contains 512^3 voxels, then we have a total amount of 64 GU. Due to the fact of the more complex spiral, illumination interruption may occur and no analytical formula can be given to calculate the GU-value. In the above described geometry reconstruction of a typical volume having 512^3 voxels we need 53.1 GU using the EPBP algorithm.

The code measured here is an optimized version which uses well known techniques like loop unrolling and subset building to achieve high cache utilization and so best performance (optimization tricks can be found in Ref. [22]). For multiprocessing (we have up to 16 cores for data processing) the code is parallelized with OpenMP directives. A more detailed timing measurement of the complete reconstruction in Table 3 shows the negligible influence of the final slice rotation step in comparison to the whole reconstruction.

To obtain an overview of the resources required we summarize the size of the weighting and look-up tables and of the data arrays in Table 4. The number of entries of the look-up tables was calculated using the following formula:

$$dN \cdot I \cdot J \cdot (KK + K). \tag{8}$$

KK is highly dependent from the algorithm used. KK represents the numbers of projections needed for reconstructing one complete slice. For example we find $KK = 632$ for the EPBP algorithm whereas the 1-Pi window reconstruction yields $KK = 128$. Note that Table 4 was generated using $K = 256$. Consequently it took us two calls to the backprojection function to reconstruct the complete volume with 512 slices.

Hyperfast spiral backprojection requires a large memory bandwidth. To analyze the situation for our architecture we conducted reconstructions with varying number of cores, or threads (Fig. 6). The Cell Broadband Engine (CBE) with 2 Cell processors running at 3.2 GHz with 2 GB Ram provided by Mercury Computer Systems, Chelmsford, USA (operating system Fedora Core 7) has a memory bandwidth of 25 GB/s and is also very efficient in this case (Table 5). The GPU values are just for comparison and from [23].

Comparing this method to high performance implementations of other reconstruction types (RayConStruct, Nürnberg, Germany) we find that the spiral backprojection, although highly optimized, is still slower than the computational intensive and hard to vectorize perspective backprojection (Table 5) on the CPU. Tests with a smaller amount of rawdata (a downsampled detector) shows a convergence of the performance from spiral backprojection towards the parallel backprojection.

Table 3 – Timing results for the hyperfast EPBP steps running on the Xeon 7300 platform using 24 threads.

Task	Time
Convolution	0.085 s
Backprojection	2.980 s
Slice rotation	0.054 s

Table 4 – Table entries and memory usage needed for reconstructing $K = 256$ slices.

	Entries	Memory usage
Weighting tables	209×10^6	800 MB
Lookup tables	209×10^6	800 MB
Raw data	116×10^6	444 MB
Volume data	67×10^6	256 MB
Sum		2300 MB

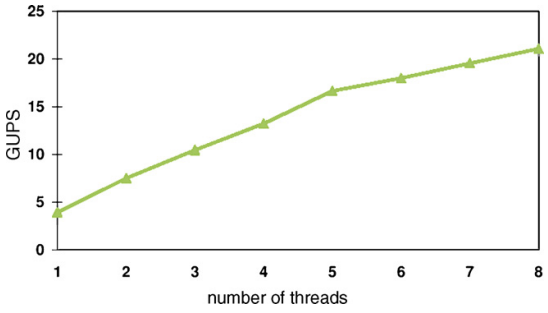


Fig. 6 – Scaling on the system using multiple threads.

Table 5 – Comparison between different reconstruction algorithms and platforms.

Architecture	Cores	Spiral BP	Perspective BP	Parallel BP
Cell (Mercury CBE)	8	23 GUPS	4.7 GUPS	21 GUPS
Cell (Mercury CBE)	16	46 GUPS	9.4 GUPS	42 GUPS
Intel Xeon 5570	8	18.7 GUPS	8.7 GUPS	51 GUPS
NVIDIA GeForce 8800 GTX	128	–	15 GUPS	13 GUPS

COMM-2960; No. of Pages 8 ARTICLE IN PRESS
COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE XXX (2009) XXX–XXX

5. Discussion

Our hyperfast backprojection approach is a general-purpose backprojector since it can be used with any spiral image reconstruction algorithm. It benefits from using the symmetries of the trajectory. The concept itself can be adapted to other trajectories of high symmetry as well. A drawback of the proposed algorithm is the requirement for circular FOVs that are centered about the rotation center. This drawback is somewhat compensated by the high reconstruction speed of the algorithm. The fact that the final slice rotation step is required may be regarded as a disadvantage. However, in combination with more sophisticated reconstruction grids, such as the hexagonal grid for example, it is an ideal setting. Computationally, the final slice rotation is costless.

The performance values shown in this paper are only an example that is valid for this specific scanner geometry and reconstruction algorithm. Our experiments have shown that slight changes of the geometry (e.g. a different cone-angle, or a different numbers of slices) may significantly change those values. The numbers cited here are the most conservative ones we observed. The performance values that we obtain for various geometry range from about 10 GUPS to up to 34 GUPS on the CPU. These significant variations between geometries and algorithms can be understood regarding the fact that memory bandwidth constitutes the major performance bottleneck, today. Different memory access patterns (e.g. by changing the spiral pitch value) may therefore have significant effects.

The most significant impact on the bandwidth bottleneck is the access to the rawdata, which is hardly predictable. The rawdata storage pattern can be improved if one desires to optimize the algorithm for a specific scanner and scan geometry.

6. Other publications

Recently other implementation for spiral CT were published (Comparisons to parallel-beam or perspective cone-beam backprojection algorithms can be found in reference [1]).

One of them using a standard Siemens geometry of 1160 projections and 672 detector channels [24]. This group uses a HP 6200 Workstation with 2 Intel Xeon 3.2 GHz processors for their backprojection. They backproject one slice with 512×512 pixels with their optimized algorithm within 1.32 s. This reconstruction has an operational count of 0.28 GU and given the speed of 1.32 s per slice we find that they achieve a performance of 0.21 GUPS.

A GPU-based spiral image reconstruction was analyzed in Ref. [25]. The authors report about reconstructions of simulated data (256 slices) and measured data (216 slices) with a pitch value of one. Each slice has 256×256 pixels and there are 360 projections (simulation) and 720 projection (measurement) per rotation to backproject. With a voxel update number of 5.6 GU for the simulation and 9.5 GU for the measurement this is a rather tiny problem. They are using two different GPUs, the best results are obtained with an NVIDIA GeForce 8800 GTX where they observe up to 1.5 GUPS. Their CPU-based reference code achieves only 0.07 GUPS.

7. Summary and conclusion

The performance of spiral cone-beam image reconstruction is improved using our hyperfast general-purpose backprojection approach. Due to using the spiral symmetry in an elegant way we achieve a high level of vectorization combined with parallelization. The approach is compatible with today's CPUs but also performs very well on the CBE.

A 11-fold performance improvement was observed when comparing this new approach to our highly optimized conventional spiral cone-beam backprojector. Comparing to other groups our hyperfast approach seems to be faster by order of magnitudes.

The image quality achievable with the new backprojector is identical to conventional reconstructions.

Acknowledgments

This work was supported by Visage Imaging GmbH, Berlin, Germany. Parts of the reconstruction software were provided by RayConStruct GmbH, Nürnberg, Germany. We thank the Intel Corporation, for providing their latest multi-core platforms.

REFERENCES

[1] M. Kachelrieß, M. Knaup, O. Bockenbach, Hyperfast parallel-beam and cone-beam backprojection using the cell general purpose hardware, Medical Physics 34 (April 4) (2007) 1474–1486.
[2] M. Kachelrieß, M. Knaup, W.A. Kalender, Extended parallel backprojection for standard 3D and phase-correlated 4D axial and spiral cone-beam CT with arbitrary pitch and 100% dose usage, Medical Physics 31 (June 6) (2004) 1623–1641.
[3] M. Kachelrieß, M. Knaup, W.A. Kalender, Multi-threaded cardiac CT, Medical Physics 33 (July 7) (2006) 2435–2447.
[4] K. Taguchi, B.S.S. Chiang, M.D. Silver, A new weighting scheme for cone-beam helical CT to reduce the image noise, Physics in Medicine and Biology 49 (June 11) (2004) 2351–2364.
[5] X. Tang, J. Hsieh, R.A. Nilsen, S. Dutta, D. Samsonov, A. Hagiwara, A threedimensional-weighted cone beam filtered backprojection (CB-FBP) algorithm for image reconstruction in volumetric CT-helical scanning, Physics in Medicine and Biology 51 (2006) 855–874.
[6] P.E. Danielsson, P. Edholm, J. Eriksson, M. Magnusson-Seger, H. Turbell, Helical cone beam scanning an reconstruction of long objects using 180 degree exposure, Patent Appl. PCT/SE 98/00029 (January 1998).
[7] M. Kachelrieß, S. Schaller, W.A. Kalender, Advanced single-slice rebinning in cone-beam spiral CT, Medical Physics 27 (4) (2000) 754–772.
[8] K. Taguchi, H. Aradate, Algorithm for image reconstruction in multi-slice helical CT, Medical Physics 25 (1998) 550–561.
[9] G.L. Larson, C.C. Ruth, C.R. Crawford, Nutating slice CT image reconstruction apparatus and method, United States Patent 5,802,134 (1998).
[10] K. Stierstorfer, T. Flohr, H. Bruder, Segmented multiple plane reconstruction: a novel approximate reconstruction scheme for multi-slice spiral CT, Physics in Medicine and Biology 47 (2002) 2571–2581.

COMM-2960; No. of Pages 8

[11] M. Defrise, F. Noo, H. Kudo, A solution to the long-object problem in helical cone-beam tomography, Physics in Medicine and Biology 45 (March 3) (2000) 623–643.
[12] R. Proksa, T. Köhler, M.T.J. Grass, The n-pi-method for helical cone-beam CT, IEEE Transaction on Medical Imaging 19 (2000) 848–863.
[13] C. Bontus, T. Köhler, R. Proksa, A quasiexact reconstruction algorithm for helical CT using a 3Pi acquisition, Medical Physics 30 (September 9) (2003) 2493–2502.
[14] A. Katsevich, Theoretically exact FBP–type inversion algorithm for spiral CT, SIAM Journal of Applied Mathematics 62 (2002) 2012–2026.
[15] F. Noo, J. Pack, D. Heuscher, Exact helical reconstruction using native conebeam geometries, Physics in Medicine and Biology 48 (December 23) (2003) 3787–3818.
[16] S. Steckmann, M. Knaup, M. Kachelrieß, Algorithm for hyperfast cone-beam spiral backprojection, in: 11th International Conference on Medical Image Computing and Computer Assisted Intervention, Workshop High-Performance Medical Image Computing and Computer Aided Intervention, 2008.
[17] S. Steckmann, M. Knaup, M. Kachelrie, High performance cone-beam spiral backprojection with voxel-specific weighting, Physics in Medicine and Biology 54 (2009) 3691–3708.
[18] E.V.R.D. Bella, A.B. Barclay, R.L. Eisner, R.W. Schafer, A comparison of rotation-based methods for iterative reconstructionalgorithms, IEEE Transactions on Nuclear Science 43 (December 6) (1996) 3370–3376.
[19] C.B. Owen, F. Makedon, High quality alias free image rotation, in: Proceedings of 30th Asilomar Conference on Signals, Systems, and Computers, Dartmouth College Computer Science, November, 1996.
[20] L. Tosoni, S. Lanzavecchia, P.L. Bellon, Image and volume data rotation with 1- and 3-pass algorithms, Computer Applications in the Biosciences 12 (6) (1996) 549–552.
[21] M. Knaup, S. Steckmann, O. Bockenbach, M. Kachelrieß, CT image reconstruction using hexagonal grids, IEEE Medical imaging Conference Record M13–277 (2007) 2074–3076.
[22] M. Knaup, M. Kachelrieß, Acceleration techniques for 2D parallel and 3D perspective forward and backprojections, in: 9th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine, 2007, pp. 45–48.
[23] M. Knaup, S. Steckmann, M. Kachelrieß, GPU-based parallel-beam and conebeam forward- and backprojection using CUDA, IEEE Medical imaging Conference Record M10–354 (2008) 5153–5157.
[24] K. Zeng, E. Bai, G. Wang, A fast CT reconstruction scheme for a general multicore pc, Journal of Biomedical Imaging 2007 (1) (2007) 1–9.
[25] W. Bi, Z. Chen, L. Zhant, Y. Xing, Accelerate helical cone-beam CT with graphics hardware, published online, in: Proceedings SPIE 6913, 2008.

5. Ergebnisse

Bereits in der Veröffentlichung in Kapitel 4 sind Ergebnisse von Benchmarktests enthalten. Diese basieren jedoch noch auf der vorherigen Generation der Intel Architektur. Die neue Nehalem Architektur gibt nicht nur den Algorithmen einen Leistungsschub, vielmehr erlaubt sie auch durch die größeren Unterschiede im Vergleich zur älteren Architektur, Rückschlüsse auf die Eignung einer bestimmten Architektur zu ziehen.

Für die Benchmarks stand uns die Intel Xeon 7300 Plattform mit vier X7460 (2,66 GHz) Prozessoren, die jeweils 6 Kerne haben, zur Verfügung. Die Größe des Caches ist 3×3 MB für den L2-Cache und 16 MB für den L3-Cache. Als Vertreter des Workstationbereichs stand uns eine mit zwei X5460 (3,16 GHz) Quadcore Prozessoren bestückte Celsius R650 Workstation der Firma Fujitsu Siemens Computers zur Verfügung. Die Größe des L2-Caches beträgt hier 2×6 MB.

Die neue Nehalem Architektur wird durch die Celsius R670 Workstation von Fujitsu Technology Solutions mit zwei Quadcore X5570 (2,93 GHz) repräsentiert. Die X5570 Prozessoren haben 4×256 kB L2-Cache und 8 MB L3-Cache. Als Betriebssystem wird Windows XP Professional x64 Edition von Microsoft Corporation verwendet.

Die Messungen für den Cell wurden auf einer CBE mit 2 Prozessoren (3,2 GHz) von der Firma Mercury Computer Systems (Chelmsford, USA) durchgeführt. Als Betriebssystem kommt hier Fedora Core 7 zum Einsatz. Als Vergleich wird noch die Leistung einer GPU basierten Rückprojektion zur Verfügung gestellt. Hierfür wurde die NVIDIA GTX 280, NVIDIA Corporation (Santa Clara, USA) verwendet. Auf dem Host Rechner läuft das Betriebssystem Windows XP Professional x64 Edition.

Die Performancewerte zeigen eine große Abhängigkeit von der gewählten Scannergeometrie. Für die Werte in Tabelle 5.3 wurde eine der von Siemens genutzten Geometrie ähnliche verwendet, da hierfür auch die Implementierung auf dem Cell Prozessor optimiert wurde. Die Parameter der verwendeten Geometrien sind in der Tabelle 5.1 zusammengestellt.

Tabelle 5.1.: Geometrien für die Messungen der Leistung.

Parameter	Vergleich der Architekturen	Siemens Sensation 64
R_{F}	600 mm	570 mm
R_{D}	450 mm	470 mm
R_{M}	400 mm	251 mm
N_{360}	512	1160
M	512	672
L	64	64
db	0,5 mm	0,5 mm
d	32 mm	32 mm

Tabelle 5.2.: Leistung der Intel-Nehalem Archtitektur.

	BP Zeit	Aufwand	Leistung	Framerate
EPBP (conv.)	31,3 s	53,1 GU	1,69 GUPS	16,4 s^{-1}
EPBP	2,84 s	53,1 GU	18,7 GUPS	180 s^{-1}
1-Pi	1,25 s	30,7 GU	24,6 GUPS	410 s^{-1}

5.1. Messung auf verschiedenen Architekturen

Für die Benchmarks werden verschiedene Rückprojektionsalgorithmen in der jeweils optimierten Fassung verwendet. Zum Einsatz kommt die neue Spiral-Rückprojektion mit gedrehten Schichten (es wurde dabei nur die Rückprojektion vermessen, die Rotation ist gemäß der Veröffentlichung in Kapitel 4 vernachlässigbar). Die 2D parallele Rückprojektion ist vom Aufbau äquivalent mit der Spiral-Rückprojektion. Der Unterschied zwischen den beiden besteht in den Zugriffsmustern auf die Rohdaten. Diese sind bei der 2D parallelen Rückprojektion wesentlich lokaler und daher kann dieser Wert als die Peak-Performance der Spirale angesehen werden. Diese These wurde evaluiert, indem in der Rückprojektion für die Spirale der Rohdatenzugriff durch einen geringeren Bereich für die Rohdaten künstlich mit einer höheren Lokalität und damit einer Rohdatenversorgung nur durch den Cache ausgestattet wurde. In Tabelle 5.2 sind die Werte äquivalent zur Veröffentlichung im Kapitel 4 dargestellt. Zum Vergleich wird in Tabelle 5.3 noch die Leistung der perspektivische Rückprojektion mit aufgeführt.

Bei einer genaueren Analyse ist zu erkennen, dass die neue Spiral-Rückprojektion deutlich von der neuen Nehalem Plattform profitiert und fast das Niveau des Cell erreicht. Im Vergleich zum Intel Xeon 5460 ist eine Verdopplung der Leistung zu verzeichnen. Dies unterstreicht die Forderung nach einer hohen Bandbreite für diese Art der Spiral-Rückprojektion.

Abseits der x86 Hardware, hin zu spezialisierter Hardware zeigt das Cell-Blade, dass sich mit einer optimierten Architektur sehr gute Werte erzielen lassen. Als Vergleich ist

Tabelle 5.3.: Vergleich verschiedener Algorithmen an Hand ihrer Leistung auf unterschiedlichen Plattformen.

Architektur	Kerne	Spiral-BP	parallele BP	perspekt. BP
Intel Xeon 5460	8	9,7 GUPS	37 GUPS	7,9 GUPS
Intel Xeon 5570	8	21 GUPS	51 GUPS	8,7 GUPS
Intel Xeon 7460	24	16 GUPS	78 GUPS	16 GUPS
Cell (Mercury CBE)	8	23 GUPS	21 GUPS	4,7 GUPS
Dual-Cell (Mercury CBE)	16	46 GUPS	42 GUPS	9,4 GUPS
NVIDIA GeForce GTX 280	$30 \cdot 8$	–	31 GUPS	24 GUPS

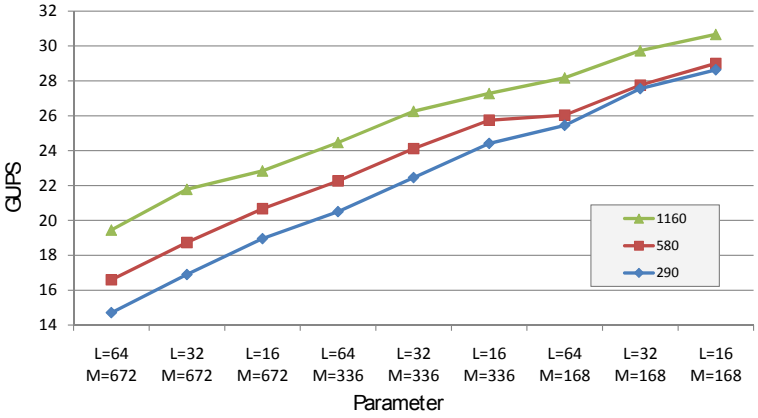


Abbildung 5.1.: Benchmarkmessungen des Spiral-Rückprojektionsalgorithmuses mit gedrehten Schichten. Dabei wurde das Detektorsampling M und L, sowie das Sampling in N-Richtung variiert. Die zugrunde liegende Geometrie entspricht der des Siemens Sensation 64.

hier noch ein Vertreter der GPU Fraktion mit aufgeführt. Eine optimierte Version der Spiral-Rückprojektion wurde hierfür nicht erstellt, da sich bereits die Implementierung der parallelen Rückprojektion als wenig effizient zeigte. Eine Eigenschaft der GPU ist es, dass sie nicht direkt auf Vektoren angewiesen ist, sondern sehr gut mit skalaren Werten arbeiten kann. Als ein gutes Beispiel ist hier die perspektivische Rückprojektion zu nennen. Diese ist nicht effizient vektorisierbar, benötigt jedoch im Vergleich zu den anderen Geometrien einen relativ hohen Rechenaufwand im Vergleich zur Speicherbandbreite. Dies ist auch gut beim Vergleich zwischen dem X5460 und dem X5570 System zu erkennen. Hierbei ist die neue Architektur bei der perspektivischen Rückprojektion nur um 10% schneller, während die Spiral- und die parallele Rückprojektion um 48% bzw. 116% an Leistung gewinnen.

Tabelle 5.4.: Ergebnisse für die Variation von N, M und L in der Siemens Sensation 64 Geometrie. Die Rekonstruktionszeit bezieht sich auf 256 Schichten.

N	M	L	GUPS	GU	Zeit in s
1160	672	64	19,5	55,9	2,87
		32	21,8	55,9	2,57
		16	22,8	55,9	2,45
	336	64	24,5	55,9	2,29
		32	26,3	55,9	2,13
		16	27,3	55,9	2,05
	168	64	28,2	55,9	1,98
		32	29,7	55,9	1,88
		16	30,7	55,9	1,82
580	672	64	16,6	27,7	1,67
		32	18,7	27,7	1,48
		16	20,7	27,7	1,34
	336	64	22,3	27,7	1,24
		32	24,1	27,7	1,15
		16	25,7	27,7	1,07
	168	64	26,0	27,7	1,06
		32	27,8	27,7	1,00
		16	29,0	27,7	0,95
290	672	64	14,7	14,0	0,95
		32	16,9	14,0	0,83
		16	19,0	14,0	0,74
	336	64	20,5	14,0	0,69
		32	22,5	14,0	0,63
		16	24,4	14,0	0,58
	168	64	25,4	14,0	0,55
		32	27,6	14,0	0,51
		16	28,6	14,0	0,49

5.2. Variation der Scannergeometrie

Einen deutlichen Einfluss auf die Spiral-Rückprojektion mit gedrehten Schichten hat die verwendete Geometrie des Scanners. Für eine komplette Betrachtung ist jedoch der zur Verfügung stehende Raum mit Parametern zu groß. Daher beschränkt sich die Untersuchung auf die von Siemens aktuell verwendete Geometrie, die seit dem Siemens Sensation 64 nahezu unverändert geblieben ist. Zwischen den Messungen wurden die Anzahl der Projektionen (N_{360}) und die Detektorabtastung (M, L) variiert.

Der Fall für $N = 1160$, $M = 672$ und $L = 64$ dient als Referenzpunkt. Dies ist der Fall, wenn ein unveränderter Datensatz rekonstruiert werden soll. Bei der Analyse der Daten mittels des Diagramms in Abbildung 5.1 fällt zunächst auf, dass ein kleinerer Datensatz nicht automatisch eine höhere Leistung des Algorithmus zur Folge hat. Eine Verringe-

rung des Samplings auf dem Detektor ergibt jedoch einen deutlichen Leistungsvorteil (Im Diagramm in x-Richtung dargestellt). Dabei hat die M-Richtung einen wesentlich höheren Einfluss auf diesen Effekt als die L-Richtung. Eine Vergrößerung der Winkelabstände, also eine Verringerung der Projektionsanzahl, bringt jedoch keinen Vorteil, da in der N-Richtung ebenso ein Subsetting für eine effektive Cache-Nutzung durchgeführt wird. Vielmehr ergibt sich durch den größeren Winkelabstand und damit den größeren Unterschied zwischen den Projektionen eine geringere Effektivität des Caches, besonders an den Randstellen der Subsets. Dass dabei jedoch die Rekonstruktionszeit im Gesamten sinkt, ist einem anderen Umstand gezollt. Mit einer Halbierung von N_{360} halbieren sich auch die notwendigen GU, weswegen unter dem Strich eine schnellere Rekonstruktion möglich wird.

Als Fazit aus dieser Teiluntersuchung lässt sich ziehen, dass ein Upsampling des Detektors vermieden werden sollte. Bei einer Implementierung für einen spezifischen Anwendungsfall muss also geprüft werden, ob eine nearest neighbor oder eine lineare Interpolation durchgeführt werden soll.

6. Medizinische Einsatzgebiete

Für den Radiologen ergibt sich bereits heutzutage ein flüssiger Ablauf der Rekonstruktion nach der Aufnahme. Selbst Rekonstruktionen in Echtzeit sind aus technischer Sicht bei den heutigen Scannern kein Problem mehr. Die Vorteile sind vielmehr in der Entwicklung zukünftiger Anwendungen zu sehen.

In zukünftigen, hochauflösenden Scannern steigt das Rohdatenaufkommen und die Rekonstruktionszeit stark an. So wird heute bereits in Nischen anstatt von Schichtbildern im Format 512×512 schon auf 1024×1024 rekonstruiert. Einhergehend hiermit steigt der Rechenaufwand für die Rekonstruktion um den Faktor 4 pro Schicht an. Soll gleichzeitig der Schichtabstand halbiert werden, steigt der Rechenaufwand sogar um den Faktor 8 an. Dies lässt sich mit dem hier vorgestellten Algorithmus wieder auf ein Niveau bringen, so dass diese hochauflösenden Rekonstruktionen bereits mit der heute verfügbaren Hardware wieder in der gleichen Zeit möglich sind.

Interessanter sind jedoch die neuen Möglichkeiten, die sich hinsichtlich neuer Algorithmen zur Rekonstruktion, wie zur Artefaktreduktion ergeben. Diese Methoden beinhalten oftmals iterative Schritte, in der in sich wiederholender Reihenfolge eine Rückprojektion und ein anschließende Vorwärtsprojektion notwendig ist. Eine Struktur allgemeiner iterativer Algorithmen ist in Abbildung 6.1 zu finden. Charakteristisch ist dabei der Initialisierungsprozess und die anschließenden Iterationen. Mit iterativ arbeitenden Algorithmen lassen sich physikalische Prozesse besser modellieren und Fehlerkorrekturen mit einbringen. Der Nachteil ist jedoch, dass im Vergleich zu einer gefilterten Rückprojektion der Rechenaufwand mit einem Faktor von $1 + 2N_{\text{Iterationen}}$ angesetzt werden muss. Eine typische Anzahl für einen praktisch einsetzbaren Algorithmus liegt zwischen einer und 16 Iterationen. Diese hängt jedoch stark vom vorhandenen Algorithmus und der Problemstellung ab.

Allgemein lassen sich die physikalischen Eigenschaften der Datenaufnahme in einer iterativen Rekonstruktion deutlich besser modellieren als in den analytischen Rekonstruktionen. Daneben zeigen iterative Rekonstruktionen auch eine geringere Anfälligkeit für mathematische Unzulänglichkeiten. Ein gutes Beispiel ist das Entstehen von Kegelstrahlartefakten in den heutigen approximativen Rekonstruktionsalgorithmen in

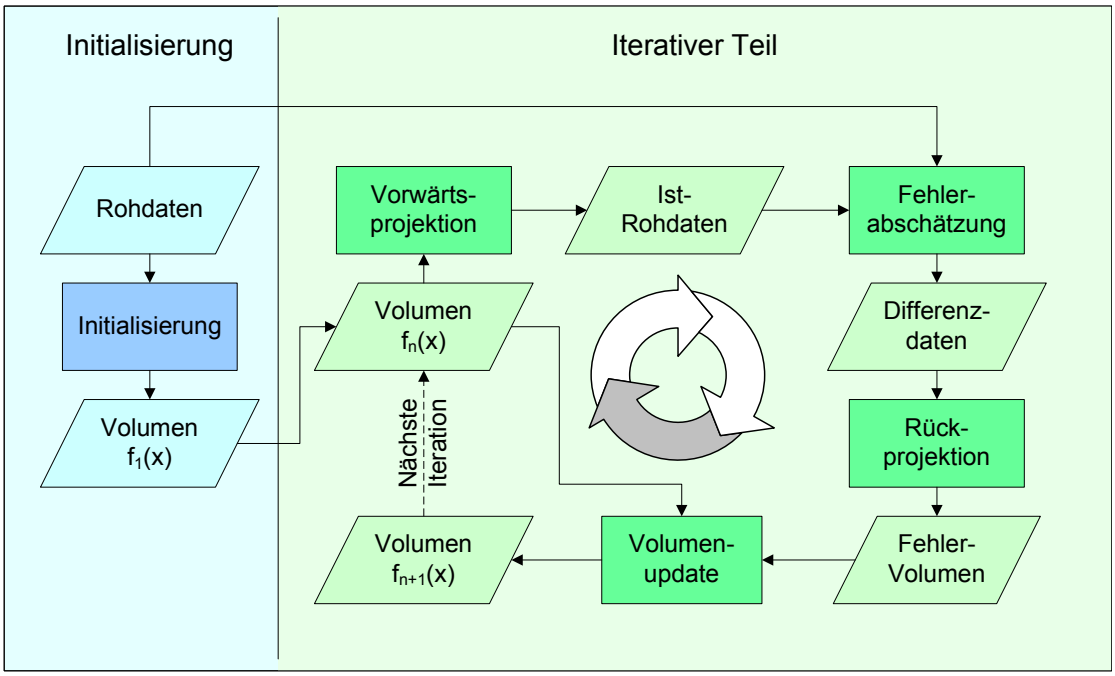


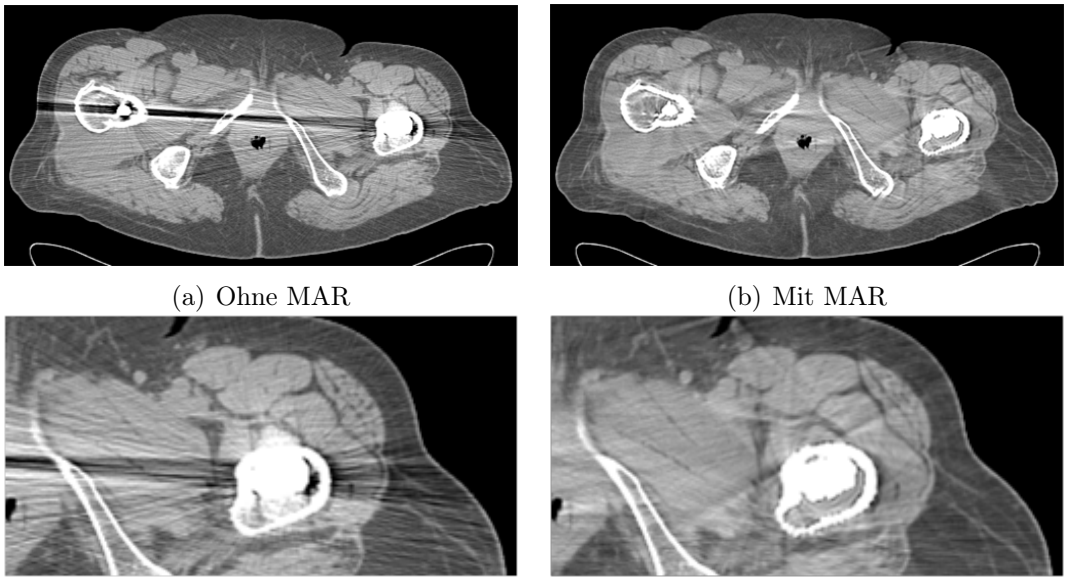
Abbildung 6.1.: Schaubild eines iterativen (Rekonstruktions-)Algorithmus. Im Allgemeinen wird von einem Startvolumen ausgegangen. Dieses kann z.B. aus einer vorherigen Rekonstruktion stammen. Im Anschluss folgen ein oder mehrere Iterationen. Jede Iteration verbessert hierbei das Volumen.

Verbindung mit den Scannern, die einen immer größeren Kegelwinkel haben. Die Spirale ist eine vollständige Trajektorie, weswegen iterative Rekonstruktionen solche Artefakte nicht zeigen. Um Rechenzeit zu sparen ist es möglich, die Kegelstrahlartefakte durch einen iterativen Schritt nach der eigentlichen approximativen Rekonstruktion zu reduzieren.

Metallartefaktreduktion

Die CT-Bilder sind oftmals von zahlreichen Artefakten überlagert. Diese erschweren dem Radiologen die Auswertung der Bilder. Dabei lassen sich Artefakte unterscheiden, die zwar die Diagnose beeinflussen und erschweren, hier sind besonders bei den modernen Scannern die Kegelstrahlartefakte zu erwähnen, während hingegen andere, wie beispielsweise Metallartefakte eine Diagnose nahezu unmöglich machen. Beide lassen sich jedoch mit modernen Verfahren soweit korrigieren, dass eine Diagnose auch in schweren Situationen möglich ist.

Metallartefakte werden in der CT durch Materialien hervorgerufen, die wesentlich dichter als der menschliche Körper sind. Oftmals entstehen sie in Prothesen und Ersatzteilen z.B. bei den Zähnen. Aber auch Clips von Eingriffen in der Cardio-CT, Rückstände von Kriegsverletzungen oder Projektile sind keine Seltenheit. Dabei machen verschiedene



(a) Ohne MAR (b) Mit MAR

(c) Ohne MAR (d) Mit MAR

Abbildung 6.2.: Metallartefaktreduktion am Beispiel eines Patienten mit einer Hüftprothese. Gemessen wurden die Daten an einem Siemens Sensation 64 bei 140 kV, dargestellt sind sie mit C=0 HU und W=500 HU.
Bildquelle: Esther Meyer in Kooperation mit Siemens Healthcare [28].

physikalische Effekte Probleme. Diese sind im wesentlichen die Strahlaufhärtung, der nichtlineare Partialvolumeneffekt, Streustrahlung und ein erhöhtes Rauschen durch erhöhte Absorption im Metall. Die Auswirkungen im Bild sind streifenförmige Artefakte und rauschähnliche Strukturen, die den unzuverlässigen Strahlen folgen. Sehr gut ist die Auswirkung auch in Abbildung 6.2 zu erkennen. In der Metallartefaktreduktion werden solche Strahlen detektiert und durch unschädliche Werte ersetzt.

Das Verfahren der Metallartefaktreduktion (MAR) basiert im wesentlichen auf einer Detektion der Metalle in einem bereits rekonstruiertem Volumen und einer anschließenden Korrektur durch Interpolation in den Rohdaten. Diese korrigierten Rohdaten werden dann wieder rekonstruiert, woraufhin das korrigierte Bild entsteht. Dieser Schritt kann zur Verbesserung der Qualität mehrmals wiederholt werden. Eine eindrucksvolle Demonstration der Möglichkeiten bietet der in Abbildung 6.2 vorgestellte Vergleich zwischen einem unkorrrigiertem Bild und dem mit einer Normalized MAR korrigierten Bild. Das Verfahren wurde in den Grundzügen von [29] vorgestellt und durch eine bessere Bildung der Normalisierung in Zusammenarbeit mit Siemens Healthcare erweitert. Dies führt zu einem Bild, in dem die Metallartefakte deutlich unterdrückt sind. Sogar Strukturen im Metall nahen Knochen sind wieder zu erkennen.

7. Diskussion

Es wurde eine neue Methode für die Rückprojektion entwickeln, die die bislang ineffizient zu vektorisierende Spiral-CT-Rückprojektion ersetzten kann. Dabei ist die neue Methode auf aktuellen Vektorrechnern hoch effizient und kann gleichzeitig flexibel mit verschiedenen Algorithmen eingesetzt werden. Erreicht wurde dies durch eine Neustrukturierung der Rückprojektion und deren Daten und dem verlassen der herkömmlichen Methoden. Dabei leistet die in der Spirale enthaltene Symmetrie einen großen Beitrag. Wenn die Schichten entsprechend mit der Spirale mit gedreht werden, wird aus der herkömmlichen Rückprojektion eine neue, die zum einen Berechnungen einspart und zum anderen vollständig vektorisierbar ist. Daraus ergeben sich zahlreiche Vorteile, der deutlichste dabei ist die Leistungssteigerung. Ausgehend von einer bereits hoch optimierten Spiral-Rückprojektion lassen sich Beschleunigungen um den Faktor 9 und mehr erzielen. Dies zeigt deutlich die Beispielmplementierung, die bei der Standardrekonstruktion für die Rückprojektion 25,7 s benötigt, wohingegen der gleiche Algorithmus mit der der neuen Rückprojektion nur noch 2,92 s auf der Intel Nehalem Architektur benötigt. Dies entspricht einem Beschleunigungsfaktor von 9. Auch auf dem Cell-Prozessor zeigt der Algorithmus eine sehr gute Leistung, die für einen einzelnen Cell-Prozessor auf dem Niveau der Intel Nehalem-Architektur mit zwei Quadcore Prozessoren liegt. Wie weiterführende Untersuchungen zeigen sind noch deutlich höhere Steigerungen möglich, die hauptsächlich durch die Geometrie des Scanners bestimmt werden. Eine große Rolle, um die größtmögliche Leistung zu erreichen, spielen auch Parameter, die an der Implementierung der Rückprojektion angepasst werden müssen. Damit wird hauptsächlich die Effizienz des Caches oder des zur Verfügung stehenden Speichers beeinflusst. Beispiele hierfür sind spezielle Subsetting Parameter, mit denen die Größe des aktuell bearbeitenden Volumens verändert werden kann. Diese können für einen bestimmten Scanner durch Benchmarkmessungen ermittelt und anschließend für die Rekonstruktionen eingesetzt werden.

Durch die Nutzung der Spiralsymmetrie und der abschließenden Drehung ergeben sich aber auch Einschränkungen hinsichtlich der Flexibilität. Durch die abschließende Drehung ergibt sich für das Field of View (FOV) eine kreisförmige Form, das zudem seinen Mittelpunkt im Drehzentrum haben muss. Für eine Rekonstruktion eines Ausschnittes

ist es also erforderlich, dass das gesamte FOV rekonstruiert wird. Dies wiegt jedoch die ohnehin sehr kurze Rekonstruktionszeit wieder auf, weswegen auch ein späteres Beschneiden nach wie vor annehmbare Rekonstruktionszeiten liefert. Weiterhin war es bis jetzt möglich, Schichten in beliebigen Abständen zu rekonstruieren. Wie schon im Kapitel 4 ausgeführt, kann der Algorithmus nur diskrete Schichten rekonstruieren. Als Basis für diese Diskretisierung dient der Tischvorschub und die Projektionsanzahl pro voller Umdrehung. Die Basis ergibt sich dann zu $\Delta n = d/N_{360}$, wobei hier vielfache oder auch ganzzahlige Teiler möglich sind, weswegen sich hier in der Praxis keine nennenswerten Einschränkungen ergeben.

Durch den hier neu entwickelten Algorithmus zur Rückprojektion können viele bereits bekannte Algorithmen in der Computertomographie profitieren. Dabei profitiert nicht nur eine bestimmte Sorte von Rekonstruktionsalgorithmen, sondern vielmehr alle, die eine Rückprojektion benötigen. Darunter fallen solche, die approximativ und mit Voxelgewichtung arbeiten wie [17, 26, 40, 41] und auch diejenigen, die mit einer Pi-Fensterung arbeiten [7, 8, 34, 42]. Daneben finden sich noch exakte und quasiexakte Algorithmen [4, 23, 24, 31], die ebenfalls mit dem neuen Ansatz deutlich beschleunigt werden können.

Besonders profitieren jedoch die iterativen Algorithmen vom neuen Ansatz zur Rückprojektion, da bei diesen die Rückprojektion in Kombination mit einer Vorwärtsprojektion mehrmals durchgeführt wird. Analog, wie in dieser Arbeit gezeigt wurde, lässt sich ebenso in der Geometrie mit den gedrehten Schichten eine Vorwärtsprojektion implementieren. Iterative Rekonstruktionen werden in medizinischen Produkten wegen dem Rechenaufwand noch nicht eingesetzt, wohl aber im akademischen Umfeld. Für diese Algorithmen rückt nun der Einsatz in der Medizin in greifbare Nähe.

A. Literaturverzeichnis

[1] Andersen, A. ; Kak, A. : Simultaneous Algebraic Reconstruction Technique (SART): A Superior Implementation of the ART Algorithm. In: *Ultrasonic Imaging* 6 (1984), S. 81–94

[2] Beekman, F. J. ; Kamphuis, C. : Ordered Subset Reconstruction for X–Ray CT. In: *Phys. Med. Biol.* 46 (2001), Nr. 7, S. 1835–1844

[3] Berkus, T. : *Iterative CT–Bildrekonstruktion*, Friedrich-Alexander-Universität Erlangen-Nürnberg, Diss., 2005

[4] Bontus, C. ; Köhler, T. ; Proksa, R. : A quasiexact reconstruction algorithm for helical CT using a 3Pi acquisition. In: *Med. Phys.* 30 (2003), Sept., Nr. 9, S. 2493–2502

[5] Böttcher, A. : *Rechneraufbau und Rechnerarchitektur*. Springer–Verlag Berlin, 2006

[6] Buzug, T. M.: *Einführung in die Computertomographie*. Springer Verlag, 2004

[7] Danielsson, P.-E. ; Edholm, P. ; Eriksson, J. ; Magnusson-Seger, M. ; Turbell, H. : *Helical cone beam scanning an reconstruction of long objects using 180 degree exposure*. Patent Appl. PCT/SE 98/00029, Jan. 1998

[8] Defrise, M. ; Noo, F. ; Kudo, H. : A solution to the long-object problem in helical cone–beam tomography. In: *Phys. Med. Biol.* 45 (2000), März, Nr. 3, S. 623–643

[9] Feldkamp, L. ; Davis, L. ; Kress, J. : Practical Cone–Beam Algorithm. In: *Journal of the Optical Society of America* 1 (1984), Jun., Nr. 6, S. 612–619

[10] Goddard, I. ; Berman, A. ; Bockenbach, O. ; Lauginiger, F. ; Schuberth, S. ; Thieret, S. : Evolution of Computer Technology for Fast Cone Beam Backprojection. In: *Proceedings of the Computational Imaging Conference* (2007), Jan., S. published online

[11] Gordon, R. : A Tutorial on ART (Algebraic Reconstruction Techniques). In: *IEEE Transactions on Nuclear Science* NS-21 (1974), Jun., S. 78–93

[12] Gordon, R. ; Bender, R. ; Herman, G. : Algebraic Reconstruction Techniques (ART) for Three–Dimensional Electron Microscopy and X–Ray Photography. In: *Journal of Theoretical Biology* 29 (1970), S. 471–482

[13] Grangeat, P. : Mathematical Framework of Cone-Beam 3D Reconstruction via the First Derivateof the Radon Transform. In: G.T. Herman, A. L. u. F. N. (Hrsg.): *Mathematical Methods in Tomography*. Springer-Verlag, Berlin, 1990

[14] Intel Corporation: *First the Tick, Now the Tock: Next Generation Intel Microarchitecture (Nehalem)*. Whitepaper, 2008

[15] Intel Corporation: *Intel® 64 and IA–32 Architectures Optimization Reference Manual*. März 2009

[16] Kachelriess, M. ; Knaup, M. ; Bockenbach, O. : Hyperfast Parallel–Beam and Cone–Beam Backprojection using the Cell General Purpose Hardware. In: *Med. Phys.* 34 (2007), Apr., Nr. 4, S. 1474–1486

[17] Kachelriess, M. ; Knaup, M. ; Kalender, W. A.: Extended Parallel Backprojection for Standard 3D and Phase–Correlated 4D Axial and Spiral Cone–Beam CT with Arbitrary Pitch and 100% Dose Usage. In: *Med. Phys.* 31 (2004), Jun., Nr. 6, S. 1623–1641

[18] Kachelriess, M. ; Schaller, S. ; Kalender, W. A.: Advanced single-slice rebinning in cone–beam spiral CT. In: *Med. Phys.* 27 (2000), Nr. 4, S. 754–772

[19] Kak, A. C. ; Slaney, M. : *Principles of Computerized Tomographic Imaging*. IEEE Press `http://www.slaney.org/pct/`

[20] Kalender, W. ; Seissler, W. ; Vock, P. : Single–breath–hold spiral volumetric CT by continuous patient translation and scannter rotation. In: *Radiology* 173(P) (1989), S. 414

[21] Kalender, W. A.: *Computertomographie*. Publics MCD Verlag, 2006

[22] Kalender, W. A. ; Seissler, W. ; Klotz, E. ; Vock, P. : Spiral Volumetric CT with Single–Breath–Hold Technique, Continuous Transport, and Continuous Scanner Rotation. In: *Radiology* 176(1) (1990), Jul., Nr. 1, S. 181–183

[23] Katsevich, A. : Theoretically exact FBP–type inversion algorithm for spiral CT. In: *SIAM Journal of Applied Mathematics* 62 (2002), S. 2012–2026

[24] Katsevich, A. : 3Pi algorithms for helical computed tomography. In: *Advances in Applied Mathematics* 36 (2006), March, Nr. 3, S. 213–250

[25] Knaup, M. ; Kachelriess, M. : Acceleration Techniques for 2D Parallel and 3D Perspective Forward- and Backprojections. In: *9th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*, 2007, S. 45–48

[26] Kudo, H. ; Noo, F. ; Defrise, M. ; Rodet, T. : New approximate filtered backprojection algorithm for helical cone–beam CT with redundant data. In: *IEEE Nuclear Science Symposium Conference Record* 5 (2003), Oct., S. 3211–3215. – ISSN 1082–3654

[27] Küveler, G. ; Schwoch, D. : *Informatik für Ingenieure und Naturwissenschaftler 2*. Friedr. Vieweg & Sohn Verlag, 2007

[28] Meyer, E. ; Bergner, F. ; Raupach, R. ; Flohr, T. ; Kachelriess, M. : Normalized Metal Artifact Reduction (NMAR) in Computed Tomography. In: *IEEE Medical Imaging Conference Record* M09-206 (2009), S. 3251–3255

[29] Müller, J. ; Buzug, T. M.: Spurious structures created by interpolation–based CT metal artifact reduction. In: Samei, E. (Hrsg.) ; Hsieh, J. (Hrsg.): *Proceedings of SPIE - Medical Imaging*, SPIE, 72581Y

[30] Munshi, A. : *The OpenCL Specification (Version: 1.0)*. Dez. 2008

[31] Noo, F. ; Pack, J. ; Heuscher, D. : Exact helical reconstruction using native cone-beam geometries. In: *Phys. Med. Biol.* 48 (2003), Dez., Nr. 23, S. 3787–3818

[32] NVIDIA: *CUDA 2.2 Programming Guide*. Mai 2009

[33] Pharr, M. (Hrsg.): *GPUGems 2*. Addison–Wesley, 2005

[34] Proksa, R. ; Köhler, T. ; Grass, M. ; Timmer, J. : The n-Pi-method for helical cone-beam CT. In: *IEEE Transactions on Medical Imaging* 19 (2000), S. 848–863

[35] Radon, J. : Über die Bestimmung von Funktionen längs gewisser Manigfaltigkeiten. In: *Berichte der mathematisch-physikalischen Kl. Sächsischen Gesellschaft der Wissenschaften* 59 (1917), S. 262

[36] Ramachandran, G. ; Lakshminarayanan, A. : Three–Dimensional Reconstruction from Radiographs and Electron Micrographs: Application of Convolution instead of Fourier Transforms. In: *Proc. Nat. Acad. Sci. USA* 68 (1971), Sept., Nr. 9, S. 2236–2240

[37] Segal, M. ; Akeley, K. : *The OpenGL Graphics System: A Specification (Version 3.1)*. März 2009

[38] Shepp, L. ; Logan, B. : The Fourier Reconstruction of a Head Section. In: *IEEE Transactions on Nuclear Science* NS-21 (1974), Jun., S. 21–43

[39] Stierstorfer, K. ; Flohr, T. ; Bruder, H. : Segmented multiple plane reconstruction: a novel approximate reconstruction scheme for multi-slice spiral CT. In: *Phys. Med. Biol.* 47 (2002), S. 2571–2581

[40] Taguchi, K. ; Chiang, B. S. ; Silver, M. D.: A new weighting scheme for cone-beam helical CT to reduce the image noise. In: *Phys. Med. Biol.* 49 (2004), Jun., Nr. 11, S. 2351–2364

[41] Tang, X. ; Hsieh, J. ; Nilsen, R. A. ; Dutta, S. ; Samsonov, D. ; Hagiwara, A. : A three-dimensional-weighted cone beam filtered backprojection (CB–FBP) algorithm for image reconstruction in volumetric CT–helical scanning. In: *Phys. Med. Biol.* 51 (2006), S. 855–874

[42] Turbell, H. : *Cone-Beam Reconstruction using Filtered Backprojection*, Linköping Universitet, Diss., 2001

[43] Tuy, H. : An inversion formula for cone-beam reconstruction. In: *SIAM Journal of Applied Mathemtics* 43 (1983), S. 546

[44] Wang, G. ; Ye, Y. ; Yu, H. : Approximate and exact cone-beam reconstruction with standard and non-standard spiral scanning. In: *Phys. Med. Biol.* 52 (2007), Mar, Nr. 6, R1–13. `http://dx.doi.org/10.1088/0031-9155/52/6/R01`. – DOI 10.1088/0031–9155/52/6/R01

B. Publikationen

Wissenschaftliche Originalarbeiten

STECKMANN, S. ; KNAUP, M. ; KACHELRIESS, M. : High performance cone–beam spiral backprojection with voxel–specific weighting. In: *Phys. Med. Biol.* 54 (2009), S. 3691–3708

STECKMANN, S. ; KNAUP, M. ; KACHELRIESS, M. : Algorithm for hyperfast cone–beam spiral backprojection. In: *Computer Methods and Programs in Biomedicine* (In press, 2009)

Kurzveröffentlichungen und Konferenzbeiträge

KNAUP, M. ; STECKMANN, S. ; BOCKENBACH, O. ; KACHELRIESS., M. : Image reconstruction using hexagonal grids. In: *IEEE Medical Imaging Conference Record* M13–277 (2007), S. 3074–3076

KNAUP, M. ; STECKMANN, S. ; BOCKENBACH, O. ; KACHELRIESS, M. : Tomographic image reconstruction using the Cell broadband engine (CBE) general purpose hardware. In: *Society of Photo–Optical Instrumentation Engineers (SPIE) Conference Series* 6498 (2007), Jan., S. 1–10

KACHELRIESS, M. ; KNAUP, M. ; STECKMANN, S. ; MARONE, F. ; STAMPANONI, M. : Hyperfast O(2048^4) image reconstruction for synchrotron–based x–ray tomographic microscopy. In: *IEEE Medical Imaging Conference Record* M06-85 (2008), Okt., S. 3810–3813

KNAUP, M. ; STECKMANN, S. ; KACHELRIESS, M. : GPU-Based Parallel-Beam and Cone-Beam Forward– and Backprojection using CUDA. In: *IEEE Medical Imaging Conference Record* M10-354 (2008), S. 5153–5157

STECKMANN, S. ; BOCKENBACH, O. ; KNAUP, M. ; KACHELRIESS, M. : Fast cone–beam spiral CT image reconstruction. In: *Radiology* 249(P) (2008), Nov., S. 693

STECKMANN, S. ; KNAUP, M. ; KACHELRIESS, M. : Algorithm for Hyperfast Cone–Beam Spiral Backprojection. In: *11th International Conference on Medical Image Computing and Computer Assisted Intervention, Workshop High-Performance Medical Image Computing and Computer Aided Intervention*, 2008

Steckmann, S. ; Knaup, M. ; Kachelriess, M. : Hyperfast General–Purpose Cone–Beam Spiral Backprojection with Voxel–Specific Weighting. In: *IEEE Medical Imaging Conference Record* M11-3 (2008), S. 5426–5433

Kachelriess, M. ; Steckmann, S. ; Knaup, M. : Hyperfast cone–beam spiral CT image reconstruction. In: *European Radiology* 19 Suppl. 1 (2009), S. S257

C. GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the

stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

1. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. 2. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement. 3. State on the Title page the name of the publisher of the Modified Version, as the publisher. 4. Preserve all the copyright notices of the Document. 5. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. 6. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. 7. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. 8. Include an unaltered copy of this License. 9. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. 10. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. 11. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. 12. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. 13. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version. 14. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section. 15. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

D. Danksagung

Ich danke ...

... meinem Doktorvater, Prof. Dr. Marc Kachelrieß, dass er mir die Möglichkeit gab, mich mit den Algorithmen der Computertomographie zu beschäftigen und mir dabei immer wieder neue und interessante Wege aufzeigte. Ohne ihn wäre diese Arbeit so nicht möglich gewesen.

... der Firma Visage Imaging für die finanzielle Unterstützung und die Möglichkeit, meine Doktorarbeit frei zu gestalten.

... Prof. Dr. Willi Kalender, dass er in Absolventen der Fachhochschulen wissenschaftliche Fähigkeiten erkennt und an seinem Lehrstuhl fördert.

... Dr. Michael Knaup und Lars Hillebrand für die (nicht) fachlichen Bürogespräche und das gießen meiner Blumen während meiner Abwesenheit.

... Dr. Robert Lapp, der mir bereits bei meiner Diplomarbeit mit Rat und Tat zur Seite stand und auch während meiner Dissertation stets ein offenes Ohr für mich hatte.

... Gertrud Heer, Andrea Gaal und Marie-Theres Reim für die Hilfe bei so mancher bürokratischen Hürde.

... Christine, Clemens, Frank, Michael, Michaela, Maria, Phillip, Rainer, Timo, ... für die angenehmen Stunden am Abend, beim Mittagessen und beim Schwimmen am Morgen :)

... Rainer für das kritische Lesen meiner Arbeit.

... den restlichen IMPlern sowie den Mitarbeitern der CT-Imaging für die kleinen oder großen Hilfen.

... meiner Freundin Nina für ihr offenes Ohr und die schönen Stunden als Ausgleich zu meiner Arbeit.

... meiner Mutter, die immer für mich da ist, wenn man sie braucht.

... auch allen anderen, die mir auf meinem Lebensweg begegnet sind und mir neue Anregungen, Ideen oder Hilfen zukommen ließen.

E. Lebenslauf

Name:	Sven Richard Steckmann
Anschrift:	Karl-Plesch-Straße 4 90596 Schwanstetten
Geburtsdatum:	24. Mai 1981 in Schwabach
Eltern:	Helmut Steckmann und Helga Steckmann
Geschwister:	Björn Steckmann
Familienstand:	ledig

Schulbildung

1987 - 1991	Grund- und Teilhauptschule I in Schwanstetten
1991 - 1994	Gymnasium Roth
1994 - 1997	Staatliche Realschule in Roth Abschluss: Mittlere Reife am 25. Juli 1997
1998 - 1999	Städtische Fachoberschule in Nürnberg Abschluss: Fachhochschulreife am 22. Juli 1999

Hochschulbildung

10/2000 - 07/2001	Studium der Werkstofftechnik Georg-Simon-Ohm Fachhochschule Nürnberg
10/2001 - 07/2006	Studium der Elektrotechnik Georg-Simon-Ohm Fachhochschule Nürnberg mit dem Schwerpunkt Daten- und Informationstechnik Thema der Diplomarbeit: *Verteilte Rekonstruktion in der Kegelstrahl-Computertomographie* Abschluss: Dipl.-Ing.(FH) am 7. Juni 2006
seit 08/2006	Promotionsstudium (Humanbiologie) Friedrich-Alexander-Universität Erlangen-Nürnberg, Institut für Medizinische Physik.

Berufserfahrung

09/2002 - 01/2003	Praxissemester Wolf Messtechnik GmbH in Schwabach Tätigkeit: Konzeptionierung und Umsetzung einer statistikbasierten Auswertung für eine Fertigungsendkontrolle in LabView
02/2004 - 07/2004	Praxissemester Inter Control KG, Nürnberg Tätigkeit: Entwicklung von Hard- und Software für ein CAN-Bluetooth Gateway zum Einsatz in der Automobiltechnik
02/2005 - 03/2005	Studentischer Mitarbeiter Inter Control KG, Nürnberg Tätigkeit: Entwurf und Implementierung einer Library von Grafikroutinen in C zur Ausgabe auf einem LCD Display
10/2005 - 07/2006	Diplomand und Mitarbeiter Firma VAMP GmbH
seit 08/2006	Wissenschaftlicher Mitarbeiter, Doktorand Institut für Medizinische Physik, Erlangen
seit 01/2008	Freiberuflicher Projektingenieur im Bereich CT-Simulation und Rekonstruktion