Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

MAGISTERARBEIT

Diploma Thesis

Evaluation und Erweiterung von DICOM fähigen Bildverarbeitungsprogrammen zum Aufbau eines medizinischen Bildverarbeitungssystemes mit DICOM Server Anbindung.

Ausgeführt am Zentrum für medizinische Physik und Biomedizintechnik der Medizinischen Universität Wien in Zusammenarbeit mit dem Institut für

Atomphysik

der Technischen Universität Wien

unter der Anleitung von

Ao Univ.-Prof. Dr. Dipl.-Ing. Peter Homolka

durch

Bsc. Martin Waitzbauer

Stroheckgasse 3/4-7
A-1090 Wien

Wien, am 16. September 2011

Diese Arbeit ist meiner Mutter gewidmet, die sich bestimmt
sehr über meinen Abschluss gefreut hätte.

Danksagung

Ich bedanke mich bei folgenden Personen: Meinem Betreuer Peter Homolka für seine Unterstützung, und seinen immerwährenden Einsatz für meine Arbeit in selbst für ihn stressigen Situationen. Meinem Kollegen Robert Leithner für Inspirationen, und Hilfe bei der Lösung einiger Probleme. Wolfgang Birkfellner, und Michael Figl für ihre Bemühungen mir in mathematischen Problemen weiterzuhelfen. Meinem Freund Theodor Adakthylos für seine Begeisterung für mein Diplomarbeitsthema, und seine MATEMATICA Kenntnisse. Meinem Freund Boris Steiner für seine grafischen Künste. Meinem Vater Wolfgang für seine immerwährende Stütze und Unterstützung, meinem Bruder Flo für sein Vetrauen in mich, und meiner lieben Freundin Michi für ihre Zuversicht und ihren Glauben an mich (sowie ihre Fotogenität). Dem Rest meiner Freunde für alles Weitere.

Inhaltsverzeichnis

1 Einleitung **9**

2 Grundlagen **10**

2.1 Bildverarbeitung - Ein Überblick . . . 11
2.2 Fourier Transformation . . . 12
2.3 Diskrete Fouriertransformation . . . 15
2.4 2-Dimensionale DFT . . . 17
2.4.1 Eigenschaften der 2 DFT . . . 17
Phasenwinkel & Powerspektrum . . . 17
Periodidizät . . . 20
Faltung . . . 20
Translation . . . 22
Rotation . . . 22
Symmetrie . . . 22
Ortsfrequenz und Auflösung . . . 22
2.5 Bildfilter . . . 25
2.5.1 Filtern im Bildraum . . . 25
2.5.2 Filtern im Frequenzraum . . . 26
Berechnung einer Filtermaske für den Ortsraum . . . 28
2.5.3 Verschiedene Filtertypen . . . 28
Tiefpassfilter . . . 28
Gaussfilter . . . 30
Hochpassfilter . . . 33
Kantendetektionsfilter . . . 33
Richtungsabhängige & Richtungsunabhängige Deteketion . . . 33
2.5.4 Methoden zur Verbesserung der Kontrastdarstellung . . . 34
Histogrammodellierung . . . 34
Histogram Equalization . . . 34

Histogrammbegrenzung . . . 34
Histogramm Matching . . . 35
Unsharp Masking . . . 35
Multy Frequency Processing (MFP) . . . 36
Vollständige Zerlegung . . . 36
2.5.5 Bildartefakte . . . 39
Bewegungsartefakte (Motion Artefacts) . . . 40
Gibbsches Ringing . . . 40
Rauschen . . . 40
Aliasing Effekt . . . 41

3 Praktische Implementation **43**
3.1 Verwendete Frameworks . . . 44
3.1.1 ITK . . . 44
Itk Daten Formate . . . 44
Besonderheiten von ITK . . . 45
3.1.2 VTK . . . 46
3.1.3 KWWidgets . . . 47
3.1.4 AVW . . . 47
AVW Daten Formate . . . 48
3.2 Implementation des MFP Algorithmus . . . 50
3.2.1 Implementation in ITK . . . 51
3.2.2 Implementation in AVW . . . 56
3.2.3 Unterschiede AVW / ITK . . . 56
3.2.4 Ergebnisse . . . 58
3.3 Slicer 3D . . . 63
3.3.1 Einleitung . . . 63
3.3.2 Installation . . . 63
Die vorkompilierte Version . . . 64
Die selbstkompilierte Version . . . 64
3.3.3 Slicer - Übersicht . . . 67
MRML . . . 67
Die grafische Benutzeroberfläche (GUI) . . . 67
3.3.4 Module . . . 69
Neue Module . . . 71
3.4 Programmieren in Slicer . . . 72

3.4.1 Systemarchitektur . . . 72
CLP Module schreiben . . . 73
Loadable Module schreiben . . . 76
3.4.2 Umsetzung des MFP Moduls . . . 79
CLP Modul . . . 80
Loadable Modul . . . 80
Programmierung . . . 81
3.4.3 Slicer Support . . . 88
3.4.4 Zukunftsaussichten & Abschließende Bemerkungen . . . 88
3.5 OsiriX . . . 90
3.5.1 Einleitung . . . 90
3.5.2 Installation . . . 91
Die vorkompilierte Version . . . 91
Die selbstkompilierte Version . . . 92
3.5.3 OsiriX - Übersicht . . . 93
3.5.4 Plugins . . . 96
Neue Plugins . . . 97
3.6 Programmieren in OsiriX . . . 98
3.6.1 Systemarchitektur . . . 98
3.6.2 Pluginarchitektur . . . 98
3.6.3 OsiriX Frameworks einbinden . . . 103
ITK . . . 103
VTK . . . 105
3.6.4 Umsetzung des MFP Moduls . . . 105
3.6.5 OsiriX Support . . . 108
3.6.6 Zukunftsaussichten und abschließende Bemerkungen . . . 111
3.7 Analyze 10.0 . . . 112
3.7.1 Einleitung . . . 112
3.7.2 Installation . . . 112
3.7.3 Analyze - Übersicht . . . 112
3.7.4 Die Basismodule . . . 114
File . . . 114
Display . . . 115
Process . . . 117
Segment . . . 118

Register 119
Measure 119
Apps 120
ITK 121
Other 121
3.7.5 Umsetzung des MFP Moduls 121
3.7.6 Analyze Support 122
3.7.7 Abschließende Bemerkungen 124
3.8 Laufzeitberechnungen & Bekannte Fehler 125
3.8.1 Slicer3 Modul 125
3.8.2 OsiriX Plugin 128
3.8.3 AVW Plugin 128

4 Conclusio und Aussichten **132**

5 Literaturverzeichnis **137**

Kapitel 1

Einleitung

Im Zuge der rasanten Entwicklung bei medizinischen Bildgebungsmodalitäten, und der entsprechenden nachträglichen Bildverarbeitung, sowie der Archivierung von Bilddaten ist international ein enormes Bedürfnis nach adäquater Software entstanden, die den individuellen Wünschen der Anwender nachkommt. Nicht selten ist aber die Verwendung einer solchen Software auch an die Frage der finanziellen Leistbarkeit geknüpft: Teure Lizenzgebühren, Schulungs & Wartungsgebühren können die Verwendung einer professionellen medizinischen Bildbewarbeitungssoftware zu einem echten Problem für Regionen machen, in denen nicht soviel Geld für das Gesundheitswesen zur Verfügung steht. Aus dieser Überlegung heraus sind in den letzten Jahren unzählige frei verfügbare, kostenlose Konkurrenzprodukte entstanden. Natürlich ist dabei nicht immer gegeben, dass die Implementation und die Funktionalität den hohen Ansprüchen der klinischen Forschung oder der medizinischen Diagnostik gerecht werden. Die Absicht dieser Arbeit war drei solcher Softwareprodukte miteinander zu vergleichen (OsiriX, Slicer 3D, Analyze) und zu evaluieren. Neben der groben Abschätzung der Funktionalität durch Evaluieren der verfügbaren Algorithmen, sowie der Dokumentation der Software, wurden im Speziellen Aspekte der Erweiterbarkeit der Software durch Plugins untersucht. In allen drei Softwareumgebungen wurde eine Methode zum Multy Frequency Processing implementiert, anhand derer die verschiedenen Stärken und Schwächen der Softwareumgebungen herausgearbeitet wurde. Dabei wurde nicht soviel Bedacht auf methodische Perfektion gelegt, als vielmehr auf die Art der Implementierung.

Kapitel 2

Grundlagen

In diesem Kapitel werden zunächst in einer Kurzübersicht grundlegende Begriffe der Bildverarbeitung, wie Bildmanipulation, Bildsegmentation und Bildregistration erklärt. Danach werden die wichtigsten formalen Grundlagen besprochen. Mit einer Kurzeinführung in die Fourieranalyse und ihrer Anwendungsgebiete wird das mathematische Grundgerüst zum Multy Frequency Processing gegeben. Anschließend werden die Wirkungsweise von Bildfiltern im Orts und Frequenzraum anhand einiger Beispiele demonstriert, sowie Methoden zur Darstellung und Verbesserung von Grauwertkontrasten vorgestellt. Zuguterletzt werden noch einige mögliche Bildfehler besprochen und es wird erklärt, wodurch sie zustande kommen, und wodurch sie sich gegenenfalls vermeiden lassen.

2.1 Bildverarbeitung - Ein Überblick

Bildverarbeitung beschreibt den Prozess, bei dem ein digitales Bild so verändert wird, dass sich der Informationsgehalt des Bildes für den Betrachter verändert oder vergößert. Im Algemeinen gibt es drei große Unterkategorien, die in der Bildverabeitung Anwendung finden:

- Bildmanipulation.
- Bildsegmentierung.
- Bildregistrierung.

Bildmanipulation bezeichnet alle Prozesse, bei denen das Eingangsbild mit rechnerischen Methoden verändert wird, d.h das Ergebnis dieses Verarbeitunsgschritts ist ein manipuliertes Bild. Manipulation beinhaltet oftmals Glättung, Schärfung, oder Kontrastanhebung und kann z.B durch Filtermasken (Absch. 2.5) oder durch Schwellwertverfahren realisiert werden. Segmentation bezeichnet den Vorgang, Bilder in Teilbereiche zu unterteilen (segmentieren). Bei der Segmentation ist also nicht mehr das Bild als solches interessant, sondern vielmehr die semantische Information, die in ihm steckt. Klassische Fragen der Bildsegmentation wären z.B: Wie ist das Bild aufgebaut? Gibt es homogene Bereiche oder Objekte, und wenn ja wie viele? Die eigentliche Segmentation kann über Kantendetektion, Schwellwertverfahren, oder verschiedene Heuristiken(z.B: Minimum Entropy)erfolgen, bzw. indem Nachbarschaftsbeziehungen zwischen Bildpunkten bewertet werden. Oftmals bilden bildmanipulierte Bilder die Basis, auf der die Anwendung von Bildsegmentations Algorithmen überhaupt erst möglich ist. Bildsegmentation bildet auch die Grundlage für darauf aufbauende Disziplinien, wie *Mustererkennung* oder *Machine Learning*, indem versucht wird, gefundene Objekte zu interpretieren und zu klassifizieren. Aufgabe der Bildregistration ist es, zwei oder mehr Bilder durch eine Transformation so miteinander in Deckung zu bringen, so dass gleiche auf ihnen befindlichen Strukturen bestmöglich übereinander liegen. Dies ist notwendig, wenn Bilder zu verschiedenen Zeitpunkten gemacht wurden, und der Aufnahmewinkel leicht unterschiedlich ist, oder wenn es durch Bewegungen zu einer Verschiebung der Bilddaten kommt. Diese Disziplin findet häufig Anwendung in der medizinischen Bildverarbeitung, da verschiedene Aufnahmemodalitäten unterschiedliche Information liefern. Dies können Bilder mit anatomischer Information (Röntgen, CT) oder Bilder mit funktioneller Information (Pet,Spect,fMRI) sein, die durch Bildregistrierung ineinander übergeführt werden können. Das entstandene Bild beinhaltet dann Informationen beider Modalitäten.

2.2 Fourier Transformation

Die Fourier Transformation, benannt nach dem französischen Matematiker, Jean Babtiste Joseph Fourier (1768 - 1830) ist eine mathematische Transformation einer kontinuierlichen Funktion in ein Orthogonalsystem, in dem die unabhängige Variable x (für den eindimensionalen Fall) in eine die Frequenz des Signals codierende Variable k übergeführt wird. Man spricht auch davon, dass ein zeitliches oder räumliches Signal durch die Fouriertransformation in den Frequenzraum überführt wird. Die Variable k gibt dabei die Frequenzen in Cycles / Range an, wobei Range die Länge des gemessenen Signals darstellt (z.B in cm, oder sec), die für x codiert. (zb x = Schwingungen/sec eines Audiosignalses in Cycles/sec, oder bei Bilddaten x = dots/inch in cycles/inch). Fourier konnte beweisen, dass periodische Funktionen, egal wie komplex sie seien mögen, einer Addition von trigonometrischen Funktionen mit verschiedenen Frequenzen und unterschiedlichen Gewichtungen, entsprechen (Abb.2.1). [1].

Die Fouriertransformation (Formel 2.1) ist eine Transformation eines örtlichen Signals $f(x)$ mit der unabhängigen Variable x in den Frequenzraum $\hat{f}(k)$, wobei k die Wellenzahl (den Wellenvektor im mehrdimensionalen Fall) darstellt.

$$\hat{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} dt \tag{2.1}$$

Die Fouriertransformation besitzt mit:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(k) e^{ikx} dt \tag{2.2}$$

auch eine eindeutige, verlustfreie Umkehrung. Durch die Verwendung von Eulers Formel

$$e^{i\varphi} = \cos\varphi + \sin i\varphi \tag{2.3}$$

ist möglich die Basisfunktion $e^{i\varphi}$ als Sinus- und Cosinusanteile auf dem Einheitskreis im Bereich $[-\varphi, \varphi]$ darzustellen. Wie man sehen kann ist die Fouriertransformation periodisch um den Ursprung und von $-\infty$ bis $+\infty$ definiert, wobei die Amplitude der Funktion mit zunehmender Wellenzahl k abfällt. Wenn $k = 0$, ist die Fouriertransformation die Summe aller kontinuierlichen Werte der Funktion $f(x)$. Da die Amplitude im Ursprung um mehrere Poten-

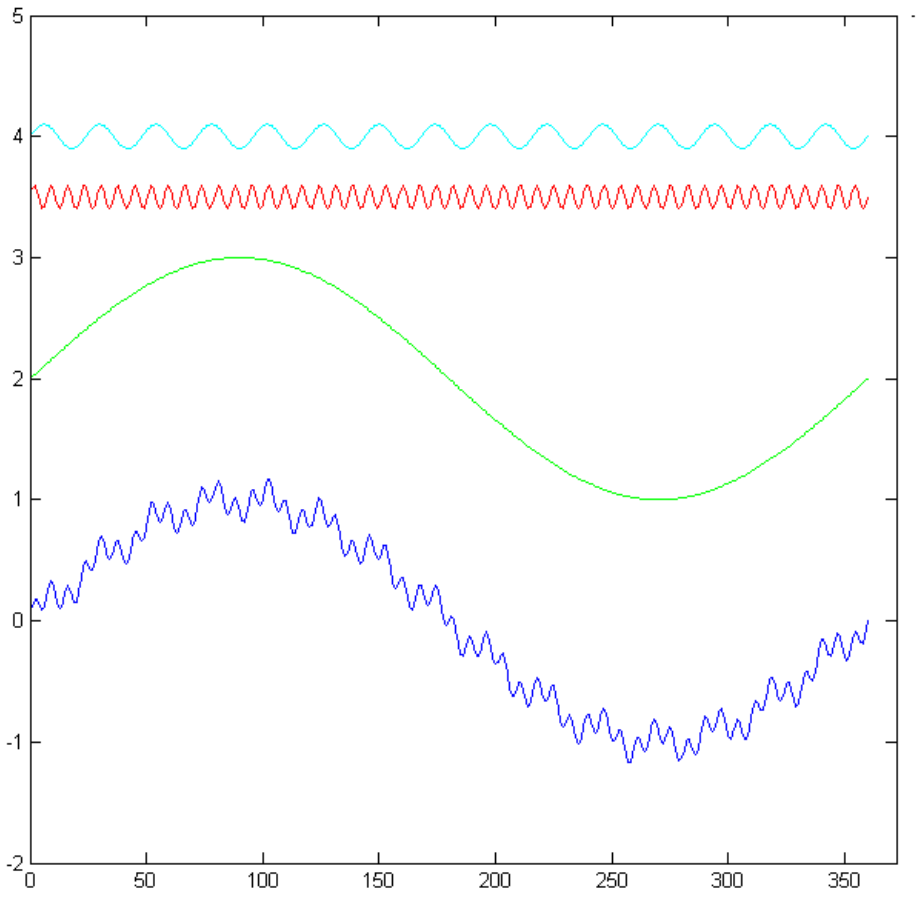

Abbildung 2.1: Dekomposition einer komplexen Funktion(unten) verschiedene Sinus und Cosinus Anteile. (darüber)

zen größer als alle übrigen sein kann, wird Die Komponente der Fouriertransformation, bei der $k = 0$ gilt, auch als *Direct Current* bezeichnet

2.3 Diskrete Fouriertransformation

Die diskrete Fouriertransformation (DFT) ist die Anwedung der Fouriertransformation auf ein diskretes Signal, dh. eine Funktion die zu gewissen Zeitpunkten ΔT von einer Impulsfunktion abgetastet wurde. Da ihre Frequenzkomponenten ausserhalb ihrers Abtastintervalls 0 sind, spricht man auch von einer bandbegrenzten Funktion [9]. Mathematisch kann gezeigt werden dass die Fouriertransformation einer bandbegrenzten Funktion nichts anderes ist, als die unendliche periodische Aneinanderreihung der Funktion. Daher reicht es aus, wenn man genau eine Periode des Signals extrahiert [9]. Die entsprechede Formel lautet:

$$\hat{f}k = \sum_{x=0}^{N-1} f(x) e^{-i2\pi kx/N} \qquad k = 0, ..., N-1 \tag{2.4}$$

wobei f eine diskrete Funktion ist. Die DFT ist anders als die Fouriertransformation nicht mehr zwischen $[-\infty, \infty]$ definiert, sondern diskret für $k = 0, ..., N-1$. N entspricht dabei indirekt der Samplingrate des Signals [9]. Die DFT besitzt ebenso wie die Fouriertransformation auch eine Inverse:

$$fx = \frac{1}{M} \sum_{k=0}^{N-1} \hat{f}(k) e^{i2\pi kx/N} \qquad x = 0, ..., N-1 \tag{2.5}$$

Sollte die Samplingrate des diskreten Signals zu niedrig sein, kommt es zu einem Verschwimmen der Perioden: das Signal verwischt und ist nicht mehr klar zu rekonstruieren. Es entstehen somit Aliasingartefakte,welche später noch erläutert werden. Ein diskretes digitalisiertes Signal ist an das Shannonsche Abtastheorem gebunden, damit es verlustfrei in das kontinuierliche bandbegrenzte Signal zurückgeführt werden kann. Shannon konnte zeigen dass für ein Signal f gelten muss:

$f_{signal} < f_{nyquist}$

und

$f_{abtast} > 2 * f_{signal}$

wobei $f_{nyquist}$ die Nyquist Frequenz darstellt, d.h die Maximalfrequenz, die in einem gemessen Signal f vorkommt. Im Klartext bedeutet das, dass das Shannonsche Abtasttheorem nur dann nicht verletzt ist, wenn das Signal mit dem Doppelten der Maximalfrequenz des Signals abgetastet wird. Da die DFT eine eindeutige, umkehrbare Transformation aus dem Orts in den Frequenzraum darstellt, gilt diese Bedingung also sowohl im Orts als auch im

Frequenzraum. Formt man obige Bedingungen um, so ergibt sich:

$$f_{nyquist} = \frac{f_{abtast}}{2}. \tag{2.6}$$

Die maximale Wellenzahl, die in einer DFT vorkommen kann ist also gleichzusetzen mit der halben Abtastfrequenz [17].

2.4 2-Dimensionale DFT

Die zweidimensionale DFT stellt die Erweiterung der DFT auf zwei Dimensionen dar. Die Wellenzahl k wird damit zum Wellenzahlvektor mit gleicher Dimension wie das Eingabesignal. Für ein Signal der Länge X x Y ergibt sich eine Abmessung der Frequenzabstände (Abstände zwischen den Wellenzahlen)

$u = \frac{1}{M\Delta X}$ und $\Delta v = \frac{1}{N\Delta Y}$

wobei ΔX und ΔY die diskreten Abtastintervalle in X / Y Richtung sind. $X\Delta X$ bzw. $Y\Delta Y$ entsprechen dabei dem Produkt aus Abtastwerten und Abtastrate für die entsprechende Dimension, also in Summe der Länge des gemessen Signals, zB (in Sekunden, oder in cm) Wie man erkennen kann, ist der Zusammenhang zwischen Frequenzabständen und der Länge des Signals umgekehrt proportional. Je länger das gemessene Signal ist, desto höher ist die Frequenzauflösung der DFT (bis zu der Obergrenze der Nyquistfrequenz) für die entsprechende Dimension.

2.4.1 Eigenschaften der 2 DFT

Phasenwinkel & Powerspektrum

Da die Fouriertransfomation aus komplexe Werten besteht, kann sie in Polardarstellung folgenderweise dargestllt werden:

$$\hat{f}(u,v) = |\hat{f}(u,v)|e^{i\phi(u,v)} \tag{2.7}$$

wobei $|\hat{f}(u,v)|$ die Stärke des Signals in Richtung des Wellenvektors e mit dem komplexen Phasenwinkel $\phi(u,v)$ angibt. Der erste Teil der obigen Gleichung wird als Powerspektrum bezeichnet. Das Powerspektrum einer DFT ist definiert als:

$$|\hat{f}(u,v)| = \sqrt{R^2(u,v) + I^2(u,v)} \tag{2.8}$$

(I = Imaginärteil , R = Realteil) und gibt die absolute Amplitudenantwort der DFT an. Im Powerspektrum sieht man welche Frequenzen mit welcher Häufigkeit vertreten sind. Ein besonders hoher Amplitudenwert bedeutet, dass diese Frequenz häufig im Bild vorkommt. Der

Abbildung 2.2: Pfeffermühle.

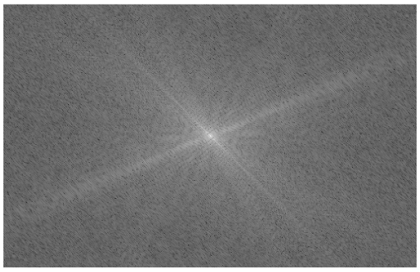

Abbildung 2.3: Powerspektrum von 2.2

höchste Wert befindet sich im Ursprung, der zugleich die niedrigste Frequenz und den durchschnittlichen Grauwert des Bildes angibt. Wenn diese Frequenz entfernt wird, entspricht das einer Abdunkelung des Bildes. Das zentrierte Powerspektrum gibt gleichzeitig einen ersten Eindruck über die Drehung des Objektes im Bild, da sich eine Drehung des Objektes auch in der Ausrichtung der Fouriertransformierten zeigt. (Gleichung 2.11) In Abb. 2.2 erkennt man die nach unten geneigte Lage der Pfeffermühle an einer entsprechenden Verstärkung der Frequenzen abseits der Hauptachsen.

Der zweite Teil von Gleichung 2.7 wird als Phasenwinkel bezeichnet. Der Phasenwinkel ist definiert als:

$$\phi(u,v) = arctan\frac{I(u,v)}{R(u,v)} \tag{2.9}$$

Für Bilder hat er die Bedeutung, dass strukturelle Information im Phasenwinkel gespeichert ist. In Abb. 2.4 wurden die zwei unteren Bilder durch Vertauschung der Phase des anderen Bildes, aber mit dem originalen Spektrum rekonstruiert. Wie man sehen kann, sind die Strukturen des anderen Bildes deutlich zu erkennen.

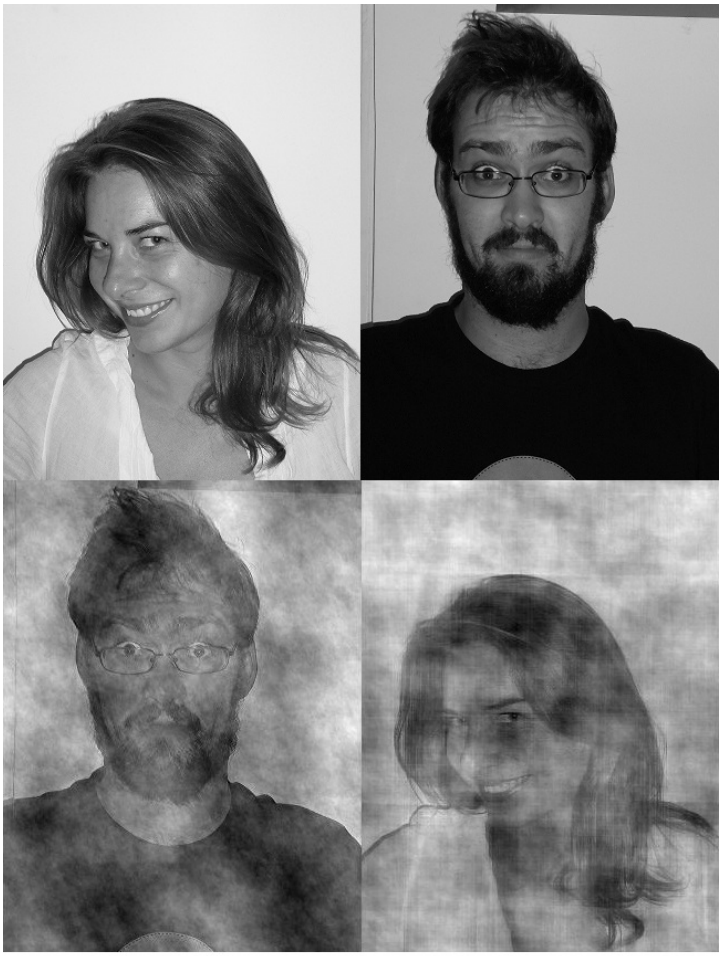

Abbildung 2.4: Obere Reihe: Originalbilder Untere Reihe: Die rekonstruierten Bilder mit einer Vertauschung der Phasenwinkel.

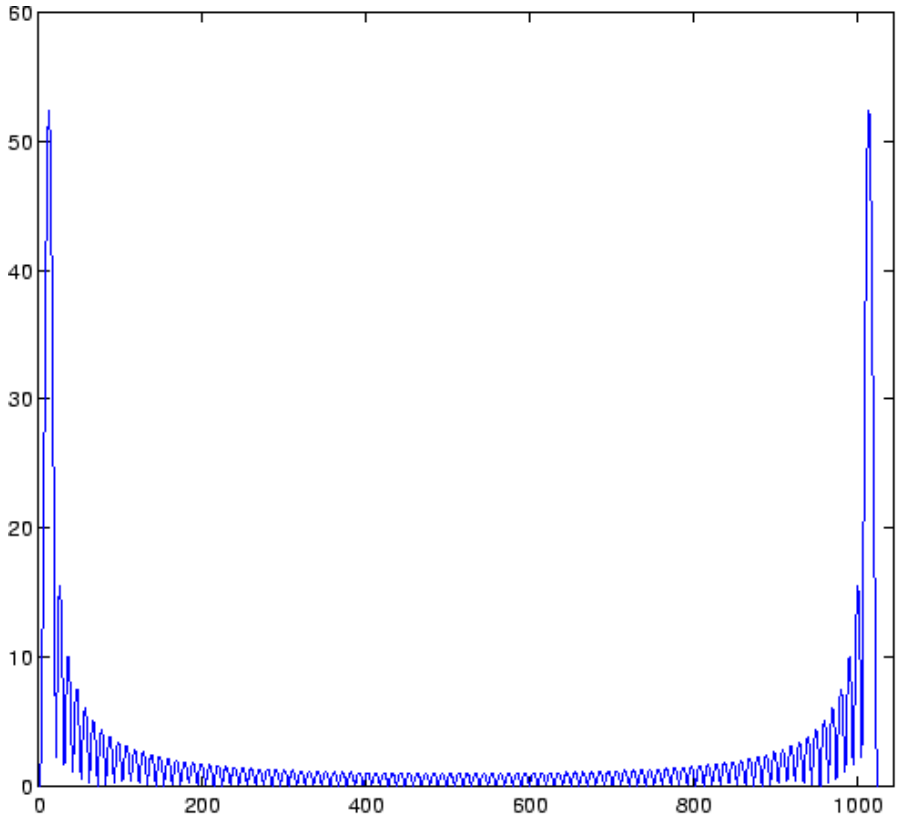

Abbildung 2.5: Fourierspektrum von eines Datensatzes mit der Länge N = 1024, die Periode ist ab N = 512 gespiegelt.

Periodidizät

Die Fouriertransformation ist auf dem Intervall $[-\infty...\infty]$definiert , die DFT nur noch im Intervall $[0,...,N-1]$, bzw. $[0...2\pi]$. Da die DFT symmetrisch um den Ursprung $k=0$ ist, hat dies zur Folge, dass die Frequenzantwort der DFT ab π, oder $\frac{N}{2}$ gespiegelt wird (Abb. 2.5). Es hat sich deshalb eingebürgert, den Ursprung entlang der X Achse zu $\frac{N}{2}$ zu verschieben (Abb. 2.5).

Faltung

Eine interessante Eigenschaft ist, dass die Faltung $f(t) \circ h(t)$ einer kontinuierlichen Funktion $f(t)$ mit einer Impulsfolge $h(t)$ im Ortsraum einer Multiplikation im Frequenzraum entspricht, und umgekehrt. (Dies hat nicht zuletzt für die Berechnungen einen enormen Vorteil, da eine Multiplikation viel einfacher und schneller durchzuführen ist als eine Faltung)

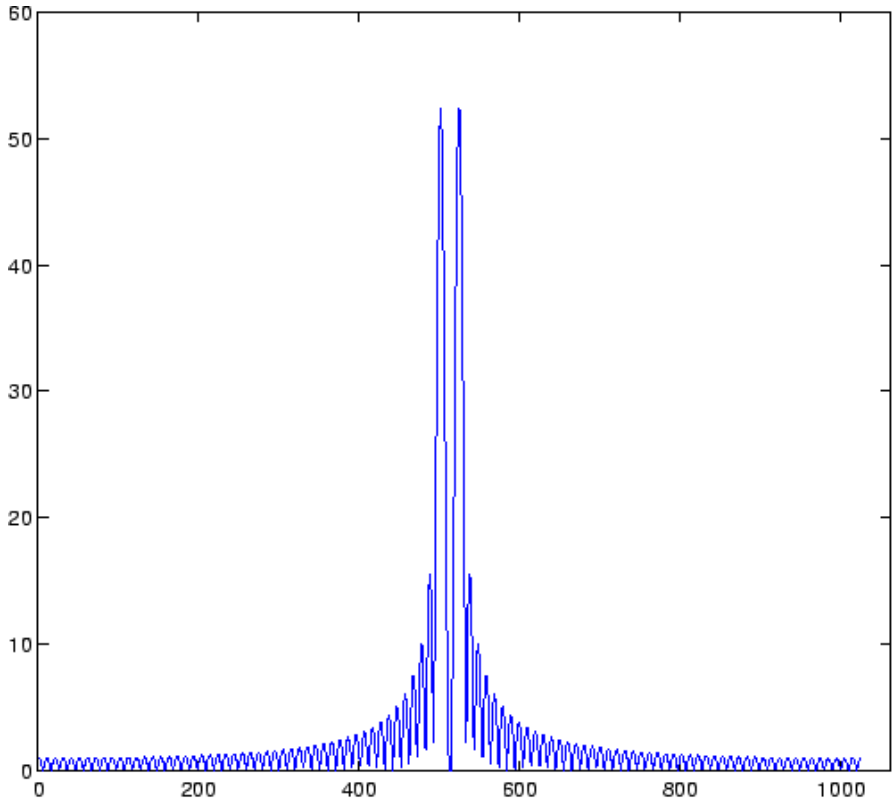

Abbildung 2.6: Entlang der X Achse verschobene Fourierdarstellung des Fourierspektrums aus Abb. 2.5

Translation

$$f(x+\alpha) \mapsto f(k)e^{k\alpha} \tag{2.10}$$

Die Translation im Bildraum bewirkt ein Veränderung der komplexen Phase. Da Strukturen im Bild hauptsächlich über die Phase definiert sind, kommt es zu sogenannten Ghosting Artfakten, also ein Durchscheinen des translierten Signals. [1]. Auf das Powerspektrum hat eine Translation im Bildraum keine Auswirkungen.

Rotation

Die DFT ist rotationsvariant. Eine Rotation der Eingangsdaten um den Winkel θ_0, bewirkt eine Rotation der Fouriertransformierten um denselben Winkel: $r = \sqrt{[x^2+y^2]}$

$$f(r,\theta+\theta_0) \Longleftrightarrow \hat{f}(\omega,\omega+\theta_0) \tag{2.11}$$

Symmetrie

Jede komplexe Funktion f kann als Summe von geraden und ungeraden Komponenten dargestellt werden. Diese Eigenschaft führte Mitte der 60er Jahre dazu, dass von den Mathematikern James Cooley und John W. Tukey die Fast Fourier Transformation Algorithmus(FFT) entwickelt wurde, der eine schnelle Implementierung des DFT Algorithmus darstellt. Die Berechnungszeit sinkt durch die FFT auf ca. $n*log(n)$, im Gegensatz zur DFT mit einer Berechnungszeit von n^2.

Ortsfrequenz und Auflösung

Ein optisches System bildet Daten der Umwelt anhand seiner Point Spread Funktion (PSF) ab. Die PSF ist eine Gütekriterium für die Abbildungsgenauigkeit, und misst dabei, wie stark ein punktförmiges Objekt durch die Abbildung verändert wird. Die PSF ist somit auch ein Maß für die räumliche Auflösung des Systems (Abb 2.7).

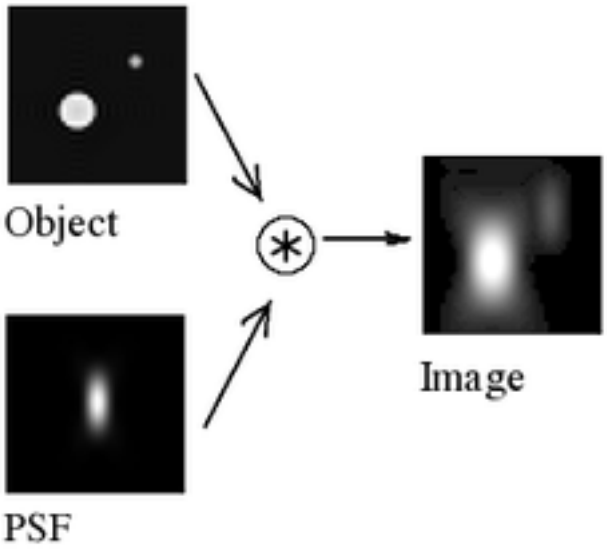

Abbildung 2.7: Faltung eines Bilddatensatzes mit einer ungenauen PSF führt zur Verwischung jedes einzelnen Bildpunktes [16]

Die Auswirkung der PSF im Frequenzraum wird über die sogenannte Modulations Transfer Funktion (MTF) festgelegt. Sie gibt an wie stark der Kontrast in Abhängigkeit zum Detailgrad im Bild dargestellt werden kann: Eine breite PSF, die dazu neigt Bildpunkte zu verschmieren, bedingt eine schnelle Abnahme der MTF zu steigenden Ortsfrequenzen, und umgekehrt. Ist der Kontrast zwischen zwei Bildpunkten nahe Null können sie auf dem Bildschirm nicht mehr getrennt voneinander dargestellt werden. Eventuelle Details verschwinden somit, und die Auflösung wird geringer. Bei jedem realen abbildenden System stehen Kontrast und Ortsfrequenz in folgendem Verhältnis: Je höher die Ortsfrequenz desto geringer die Antwort der MTF. Auflösung kann auf verschiedene Arten definiert werden, etwa in Pixeln, oder in DPI bei Druckern. Bei medizinischen Bilddaten hat sich der Begriff Linienpaare / mm eingebürgert. Ein Linienpaar stellt dabei immer eine weiße neben einer schwarzen Linie dar. Je feiner die Auflösung eines optischen Systems, desto mehr Linienpaare /mm können dargestellt werden. (Abb 2.8)

Der Begriff der der Linienpaare / mm ist auch als Ortsfrequenz bekannt, wobei die Ortsfrequenz mit zunehmenden Linienpaaren / mm steigt. Die Frequenzen die bei der Fourieranalyse gewonnen werden kodieren Detailgrade im Bild. Der geringste Detailgrad (k = 0) befindet sich im Ursprung. Dies sind alle Strukturen, die sich nicht schnell ändern, und die Grundstruktur des Bildes ausmachen. Um ein Beispiel zu nennen: Bei einem Bild einer grünen Wiese mit blauem Himmel, entspräche dies der Wiese und dem Himmel. Je weiter man

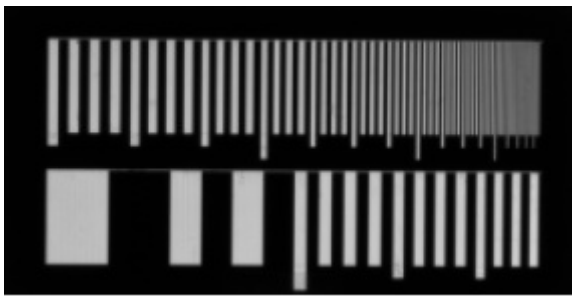

Abbildung 2.8: Phantom zur Messung von Linienpaaren [3]

sich im Fourierraum vom Ursprung entfernt desto höhere Frequenzkomponenten kodieren immer mehr Details im Bild. Auf dem Bild mit der Wiese würden immer mehr Details ersichtlich werden, wie z.B. Blumen. Bei digitalisierten Signalen entspricht die höchste darstellbare Ortsfrequenz der Nyquistfrequenz, wie bereits weiter oben festgestellt wurde. Wenn man die Größe des Bildes in mm kennt (Im Dicom Header findet sich dazu der Eintrag [0028,0030 Pixelspacing]) ergibt sich für die Berechnung der Linienpaare / mm folgende Formel:

$$Linienpaare_{mm} = \frac{\frac{1}{Pixelspacing}}{2} \tag{2.12}$$

N (x-1,y-1)	D (x,y-1)	N (x+1,y-1)
D (x-1,y)	P (x,y)	D (x+1,y)
N (x-1,y+1)	D (x,y+1)	N (x+1,y+1)

Abbildung 2.9: Filtermaske mit 8-ter Nachbarschaft [15]

2.5 Bildfilter

Bildfilter können in ihrer Funktionsweise mit der PSF eines bildgebenden Systems verglichen werden: Sie bilden das Eingangssignal entsprechend ihrer Eigenschaften auf ein Ausgangssignal ab. Filter können den Kontrast anheben, Kanten verstärken, Strukturen glätten, oder Strukturen durch einen Schwellwert von einander trennen, um nur einige zu nennen. Bildfilter lassen sich im Bildraum und Frequenzraum erzeugen und anwenden.

2.5.1 Filtern im Bildraum

Eine Faltung eines Bildfilters im Ortsraum wird berechnet, indem eine sogenannte Filtermaske, welche die PSF darstellt, mit dem Zentralpixel P über jedes Pixel des Eingangsbildes gelegt wird. Das neue resultierende Pixel P' an entsprechender Stelle ist die Faltung der Filtermaske mit dem entsprechend darunterliegendem Bildausschnitt. Filtermasken sind meist quadratisch, können aber eine beliebige Form aufweisen, im Falle einer quadratischen Filtermaske spricht man häufig von 4er oder 8er Nachbarschaft. Von 4er Nachbarschaft spricht man , wenn ein Pixel P seine 4 horizontalen , vertikalen Nachbarn D für die Berechnungen einbezieht und von 8er wenn zusätzlich noch die 4 diagonale Nachbarn N hinzugezogen werden (Abb.2.9).

Dies macht klar, dass kernelbasierte Filteroperationen nur im Bereich des Bildes angewendet werden können, wo der Bildrand mindestens eine halbe Kernelbreite vom Kernelzentrum entfernt liegt (Abb. 2.10). Innerhalb dieses Randes müsste das Kernel auf Daten zugreifen, die nicht existent sind, um eine vollständige Berechnung durchzuführen. Gängige Problemlösungen für diesen Fall sind z.B: die Verkleinerung des Bildes von $F(N,M)$ auf $F(N-\frac{kernel}{2},M-$

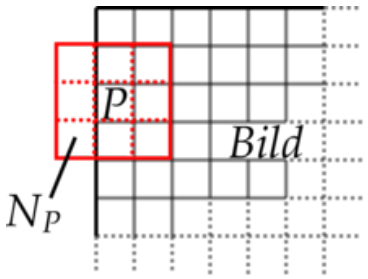

Abbildung 2.10: Randproblem bei der Verwendung von Filtermasken [15]

$\frac{kernel}{2}$, oder mit der Ergänzung der Filterdaten durch auffüllen mit Nullen. Dieses sogenannte Zero - Padding verwischt somit die Daten an den Filterrändern. Da die Berechnung eines Pixels einer Multiplikation der gesamten Filtermaske mit der Kernelumgebung des Pixels entspricht, und diese Multiplikation insgesamt mindestens $N - \frac{kernel}{2} * M - \frac{kernel}{2}$ durchgeführt werden muss, ist die Filterung im Ortsraum extrem rechnungsintensiv. Mit steigender Größe der Filtermaske steigt die benötigte Zeit exponentiell an.

2.5.2 Filtern im Frequenzraum

Die Anwendung von Filtern im Frequenzbereich bedeutet eine Modifikation des Fourierspektrums, indem Frequenzkomponenten abgschwächt oder angehoben werden. Nach der Rücktransformation in den Ortsraum sind diese Strukturen auch im Bild angehoben oder abgeschwächt. Auf diese Weise kann man etwa hochfrequentes Rauschen durch den Einsatz eines geeigneten Tiefpassfilters, entfernen, oder Bilder durch einen Hochpassfilter schärfen. Abb 2.11 zeigt dass das Originalbild (a - links oben) durch ein periodisches Rauschen einer bestimmten Frequenz gestört wird. Die Darstellung im Powerspektrum macht deutlich wo die Störfrequenzen zu finden sind (b - rechts oben). Die Filtermaske (c - links unten) bewirkt ein Ausblenden dieser Frequenzen, und somit eine Restaurierung des rücktransformierten Bildes (d - rechts unten). Durch diesen Verarbeitungsschritt wird aber nicht nur das Rauschen unterdrückt, da die Filtermakse alle Frequenzen in alle Raumrichtungen unterdrückt. Es kann somit zu einem Unschärfe effekt kommen, wenn in der entsprechenden Frequenzkomponente strukturelle Bildinformationen vorhanden sind. Die Konstruktion eines Bildfilters im Frequenzraum muss sich also immer danach ausrichten, wo relevante Frequenzen im Bild vorhanden sind.

Ein großer Vorteil von Filteroperationen im Frequenzraum ist, dass sich eine Faltung einer Filtermaske mit einem Bild zu einer Multiplikation vereinfacht. Dies ist speziell bei großen

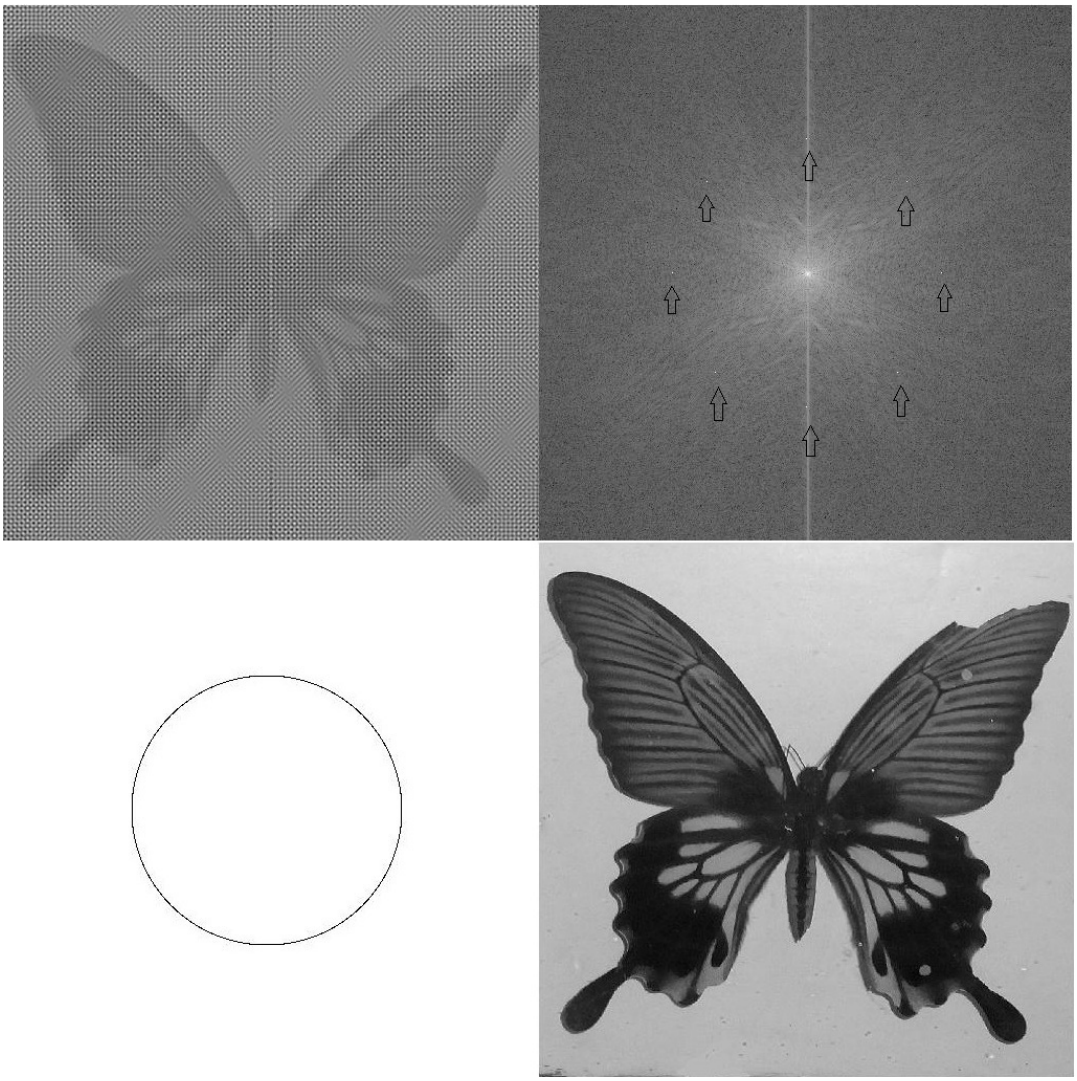

Abbildung 2.11: Bildrestauration durch geeignete Filtermaske im Frequenzraum

Bilddaten von enormer Wichtigkeit, da hier die Berechnungszeit drastisch gesenkt werden kann.

Berechnung einer Filtermaske für den Ortsraum

Filtermasken für den Ortsraum werden berechnet, indem eine kontinuierliche Filterformel für einen Filtermaskenmittelpunkt, und dessen umliegende Nachbarpunkte berechnet wird. Im Falle des Gaussfilters mit der Formel

$$f(x,y) = e^{\frac{-x^2-y^2}{2*\sigma^2}} \tag{2.13}$$

berechnen sich die Koeffizienten für eine 3 x 3 Filtermaske F, mit $\sigma = 0.85$

$$\begin{pmatrix} F(-1,-1) & F(0,-1) & F(1,-1) \\ F(-1,0) & F(0,0) & F(1,0) \\ F(-1,1) & F(0,1) & F(1,1) \end{pmatrix} \rightarrow \begin{pmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{pmatrix}$$

Wird eine Normalisierung angewendet, so wird der rechte Teil der obigen Abbildung noch zusätzlich mit der Summe aller Koeffizienten gewichtet:

$$F = \frac{1}{4} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Der Gaussfilter, genauso wie der Binomialfilter gehören zur Klasse der linearen Filtertypen, d.h ihre Filtermaske entsteht als Multiplikation zweier Funktionen $F_1 = (1,2,1)$ und $F_2 = (1,2,1)$'. Jede dieser Masken einzeln hintereinander auf ein Bild angewendet, liefert dasselbe Ergebnis wie die zusammengesetzte Filtermaske (ohne Skalierungsfaktor). Nichtlineare Filtertypen benötigen für die Berechnung lokale Information um das zentrierte Pixel (z.B Median Filter).

2.5.3 Verschiedene Filtertypen

Tiefpassfilter

Tiefpassfilter sind Bildfilter, die die hohen Frequenzen des Bildes unterdrücken und tiefe passieren lassen. Das Ergebnis einer solchen Filteroperation ist ein unscharfes, geglättetes Bild.

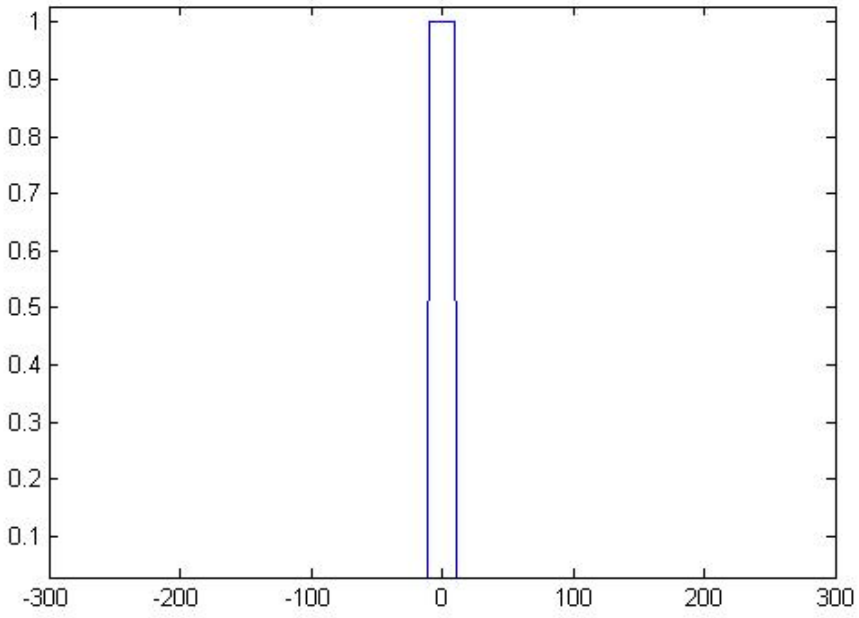

Abbildung 2.12: Idealer Tiefpassfilter

Tiefpassfilter werden üblicherweise eingesetzt, um Rauschen aus dem Bild zu entfernen und können sowohl im Orts als auch im Frequenzraum realisiert werden.

Idealer Tiefpassfilter

Der Ideale Tiefpassfilter, oder Box Filter besitzt eine Flanke mit unendlichem Anstieg, und grenzt dabei alle Frequenzen aus, die oberhalb seiner kritischen Frequenz liegen. Wird ein Box Filter im Frequenzraum definiert, so entstehen durch den unendlichen Flankenanstieg bei der Rücktransformation in den Ortsraum Über und Unterschwingungen $(Over/Undershoot)$, welche zusätzliche Bildkanten erzeugen und Bildartefakte darstellen (Ringing). Wird der Box Filter im Ortsraum definiert, so besitzt sein Faltungskern folgendes Aussehen (für eine 3 x 3 Filtermaske)

$$F = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Dies entspricht einer simplen Mittelung aller Bildwerte im Filterkernel. Es kann jedoch gezeigt werden, dass diese Mittelung nicht für alle Frequenzen erzielt werden kann.

$$\begin{pmatrix} & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & 1 & 1 & 0 & 1 & 1 & 0 & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & 1 & 0 & 1 & 0 & 1 & 0 & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{pmatrix} \bullet A = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \frac{2}{3} & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{pmatrix}$$

In obiger Abb. werden Strukturen von einem Pixel Breite durch die Anwendung des Box filters mit zusätzlichen Kanteninformationen versehen. Dies entsteht durch die Übertragungsfunktion des Boxfilters im Fourierraum, die Periodische Schwingungen gegen Unendlich enthält (Abb. 2.13) Dies führt somit zu einer Verstärkung von hohen Frequenzen, weshalb es zu den oben beschriebenen Ringing Artefakten, sowohl im Orts als auch im Frequenzraum kommt. Boxfilter sind in der Regel nicht von elektrischen Bauteilen realisierbar, und können nur virtuell am Computer berechnet werden.

Gaussfilter

Der Gauss Filter ist ein Filter mit gewichtetem Mittelwertprofil, entsprechend dem Aussehen einer gausschen Glockenkurve (Abb. 2.14) Die entsprechende Formel lautet:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2) \tag{2.14}$$

Dies entspricht der Standardformel für die Gaussche Normalverteilung mit Mittelwert μ, Standardabweichung σ, und unabhängiger Variable x. Im zweidimensionalen Fall wird aus $x-\mu$ die Euklidische Distanz d:

$$d = \sqrt{(x-\mu)^2 - (y-\mu)^2} \tag{2.15}$$

Der Normierungsfaktor $\frac{1}{\sigma\sqrt{2\pi}}$ normiert alle Gausskurven auf die gleich Fläche. Da der Gaussfilter asymptotisch gegen 0 geht, 0 jedoch nie erreicht, bedeutet dass für den Frequenzraum, dass es keine kritische Grenzfrequenz gibt, ab der die Frequenzen abgeschnitten werden.

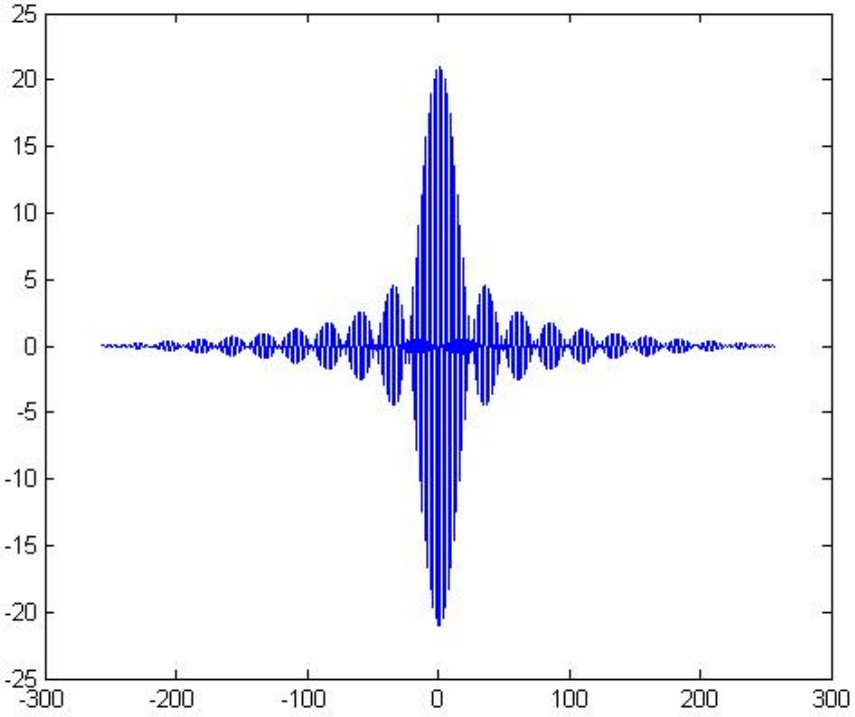

Abbildung 2.13: Fouriertransformierte des Boxfiltes aus Abb. 2.12

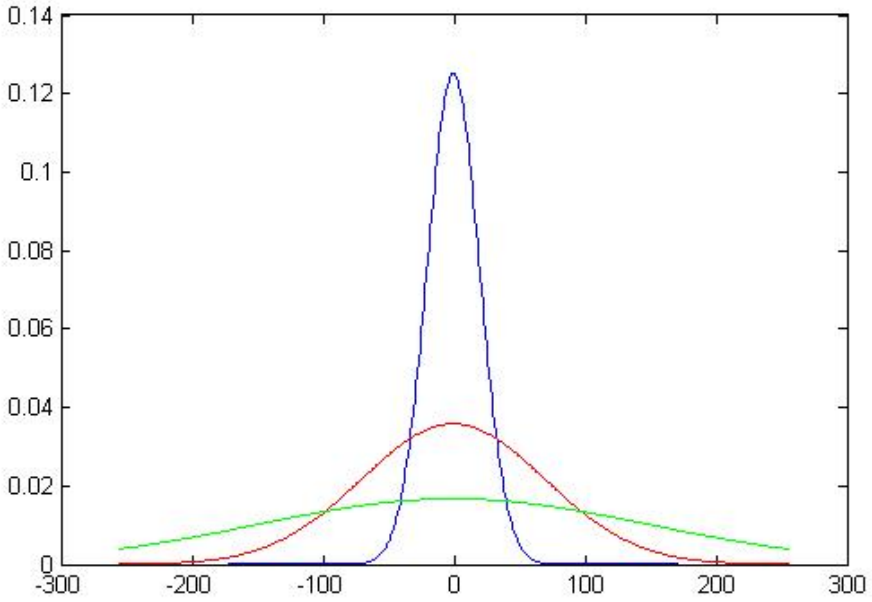

Abbildung 2.14: Normierte Gausskurven. Die Ausdehnung der Kurven ändert sich, aber ihre Fläche bleibt immer gleich.

Jede Faltung mit einer Gaussmaske enthält somit immer alle Frequenzen des gesamten Bildes, allerdings in immer kleiner werdenden Intensitäten, sodass diese für den visuellen Eindruck vernachlässigbar werden. Da der Gaussfilter keine Unstetigkeitsstellen wie der Box Filter besitzt, und seine Flanken 'glatt abrollen', kommt es bei Gauss Filtern auch zu keinem Auftreten von Ringing Artefakten. Eine weitere positive Eigenschaft des Gaussfilters ist, dass er durch eine Fourier Transformation zwar seine Form in Bezug auf die Standardabweichung σ verändert, aber sein Aussehen noch immer einer Normalverteilung folgt. Im Ortsraum wird der Gaussfilter mittels folgender Filtermaske definiert (3 x 3 Filtermaske):

$$F = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Wie man sehen kann, besitzt der Gaussfilter einen Faltunsgkern mit gewichtetem Mittelwertsprofil. Dies bewirkt eine stärkere Erhaltung von lokalen Strukturen, als z.B. durch die Anwendung des Boxfilters.

Hochpassfilter

Hochpassfilter sind das Analogon für hohe Bildfrequenzen: Sie unterdrücken niedrige Frequenzen und lassen hohe passieren. Durch Anwendung eines Hochpassfilters treten die Bilddetails stärker hervor und Kanten werden angehoben. Leider tendieren Hochpassfilter auch dazu das Rauschen im Bild zu verstärken: Homogene Bereiche im Bild verschwinden, der Gesamteindruck wird härter, und das Bild körniger.

Kantendetektionsfilter

Da sich Kanten durch sprunghafte Veränderungen der Bildfunktionswerte beschreiben lassen, basieren viele Hochpassfilter auf dem Ansatz die Bildfunktion in x, y Richtung abzuleiten. Kantendetektionsfilter lassen sich in richtungsabhängige und richtungsunabhängige Filter einteilen.

Richtungsabhängige & Richtungsunabhängige Deteketion

Von richtungsabhängiger Kantendetektionsfiltern (Prewitt, Sobel-Operator) spricht man, wenn ihre Filermasken nur Kanten in eine bestimmte Raumrichtung betonen. In fogender Grafik erkennt man die Richtung der Ableitung an der Ausrichtung der Nullen in der Faltungsmaske:

$$\begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix} \nearrow \quad \begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix} \nwarrow \quad \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \uparrow \quad \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \rightarrow$$

Somit müssen mehrere Filtermasken miteinander kombiniert werden, um Kanten in alle Raumrichtungsen zu erkennen. Eine richtunsgunabhängige Detektion gewinnt man durch den Absolutbetrag des Gradienten in jedem Bildpunkt, oder oder durch Berechnung der zweiten Ableitung im jedem Bildpunkt (Laplace Operator).

2.5.4 Methoden zur Verbesserung der Kontrastdarstellung

Die Sichtbarkeit von Objekten in Bilddaten ist immer an die Frage der Darstellung des Kontrastes gebunden. Wie bereits weiter oben festgehalten wurde , nimmt der Kontrast mit zunehmender Ortsfrequenz ab. Je nachdem also welche Strukturen betrachtet werden, ist es notwendig den Kontrast so einzustellen, dass das verfügbare Grauwertspektrum komplett ausgenutzt werden kann. Dies kann für digitale Bilddaten in den meisten Bildverarbeitungssoftwaren individuell vom Benutzer , über die Wahl der Gamma Kurve, und entsprechende Veränderung des Bild-Histograms gemacht werden. Die Gamma Kurve stellt eine Transformationsfunktion der gemessenen physikalischen Größen der jeweiligen Modalitäten auf das Grauwertspektrum des Monitors dar. (Die tatsächlich darstellbaren Graustufen sind dabei natürlich abhängig von den Möglichkeiten des Monitors diese darzustellen). Im Folgenden sollen einige Methoden zur Verbesserung des Kontrast besprochen werden.

Histogrammodellierung

Diese sind jene Kategorie von Methoden, die das Histogramm des Bildes bearbeiten um den Kontrast anzupassen. Ziel der Histogrammmodelierung ist es, immer den gesamten Bereich der Grauwertskala, von Schwarz bis Weiß auszunutzen. Manche dieser Methoden sind automatisch, andere bedürfen der Festlegung von Grenzbereichen.

Histogram Equalization

Dabei wird versucht die Grauwerte des Histogramms entsprechend einer Gleichverteilung umzumodelieren. Histogramm Equalization ist eine Möglichkeit den Kontrast zu erhöhen, wenn große Bereiche des Bildes im selben Grauwertbereich anzufinden sind. Der Vorteil des Histogram Equalization ist, dass es ein vollautomatisches Verfahren ist, das ohne irgendeinen Eingabeparameter verwendet werden kann.

Histogrammbegrenzung

Histogrammbegrenzung wird angewandt, wenn das Histogramm eine breite Verteilung der Werte aufweist, die Amplitude seine Verteilungsfunktion jedoch auf einen bestimmten Bereich konzentriert ist. Dabei muss eine ober und eine Unterschwelle angegeben werden,

ober bzw. unter der alle Werte auf weiß bzw. Schwarz gesetzt werden. Die dazwischen befindlichen Werte werden mittels einer linearen Interpolationsfunktion auf den neuen Grauwert gesetzt. In der Medizin wird diese Technik als Fensterung bezeichnet.

Histogramm Matching

Das Histogramm Matching benötigt ein Referenzbild, dessen Histogramm als Referenz für ein zweites Histogramm dient. Dabei wird versucht die Verteilung des zu bearbeitenden Histogramms auf die Verteilung des Referenzhistogramms anzupassen. Histogramm Matching dient also dazu, zwei Bilder, zumindest von der Verteilung der Helligkeitswerte, gleich aussehen zu lassen.

Unsharp Masking

Unsharp Masking ist eine oft angewandte Methode, mit der feine Strukturen im Bild betont werden können, wodurch ihr Kontrast verbessert werden kann. In der Praxis gibt es mehrere Ansätze die konzeptuell allerdings alle äquivalent sind:

1. Ein Bild wird tiefpassgefiltert.
2. Das entstandene Bild wird vom Original abgezogen.
3. Das Subtraktionsbild wird zum Original mit dem Gewichtungsfaktor k hinzuaddiert.

Das Subtraktionsbild beinhaltet alle Frequenzen, die zwischen dem Originalbild und dem geglätteten Bild, der Unsharp Mask liegen. Bei einem Verstärkungsfaktor $k = 1$ ergibt sich wieder das Originalbild, bei $k > 1$ werden die Strukturen des Differenzbildes hervorgehoben. Obige Methode von Unsharp Masking hat auch Nachteile [7]:

1. Große Werte für k erzeugen künstliche Kanten in Bereichen, in denen im Vorhinein ein hoher Kontrast vorhanden war.
2. Das Bildrauschen nimmt bei hohen Werten für k zu.
3. Objekte mit diffusen Grenzbereichen profitieren nur schlecht von Unsharp Masking.
4. Hohe Werte von k verschieben das Histogramm in Richtung des Subtraktionsbildes.

Eine Möglichkeit Fehler, bei einem hohen Verstärkungsfaktor zu vermeiden ist es die Unsharp Mask amplitudenmoduliert in das Ergebnisbild einfließen zu lassen: Dabei werden hohe Werte der Unsharp Mask verhältnismäßig abgeschwächt, und niedrige entsprechend stärker gewichtet.

Multy Frequency Processing (MFP)

Multyfrequency Processing (MFP) ist eine Methode zur individuellen Kontrastanhebung verschieden großer Strukturen in Bildern. Ähnlich wie die verwandte Methode Unsharp Masking [7] beruht MFP darauf, das Eingangsbild in seine spektralen Komponenten zu zerlegen, diese zu gewichten und wieder zusammenzufügen. Anders als Unsharp Masking jedoch, welches hauptsächlich zur Kantenschärfung verwendet wird, wird das Eingangsbild in beliebig viele, und nicht nur zwei Spektralbänder zerlegt. Das Ergebnis ist ein Bild, bei dem bestimmte Ortsfrequenzen (Strukturmerkmale des Bildes) stärker hervorgehoben, bzw abgeschwächt werden können. So können z.B: grobe Strukturmerkmale, die eine hohe Amplitude im Fourierraum haben, und einen hohen Kontrast besitzen, abgeschwächt werden. Feinere Objekte mit kleinerer Amplitude können verstärkt werden, und so wiederum deren Kontrast erhöht werden. Eine Zerlegung in spektrale Komponenten muss dabei so gewählt werden, dass sich alle Bandbilder mit einer neutralen Gewichtung von 1 wieder komplett zum Originalbild zusammenfügen lassen.

Vollständige Zerlegung

Ein Bild kann mittels Anwendung mehrerer Bandpassfilter in Frequenzbänder zerlegt werden, welche unterschiedliche Strukturinformationen enthalten (Abb, 2.15). Alle Bänder müssen für eine vollständig spektrale Zerlegung den gesamten Wellenraum mit Wellenzahlen von $k = 0...N-1$ $N = Cycles/Range\ des\ gesamten\ Bildes)$ enthalten. Das Originalbild ergibt sich dann durch eine Addition aller einzelnen Bänder, wenn

1. Keine Rundungsfehler bei der Berechnung der Bänder durch Hin und Rücktransformation in den Fourierraum gemacht werden.
2. Es zu keinen Bildfehlern bei den Bandfiltern kommt (z.B: Ringing).
3. Die Filterfunktionen sich auf 1 ergänzen, und die MTF somit wieder komplett herstellen.

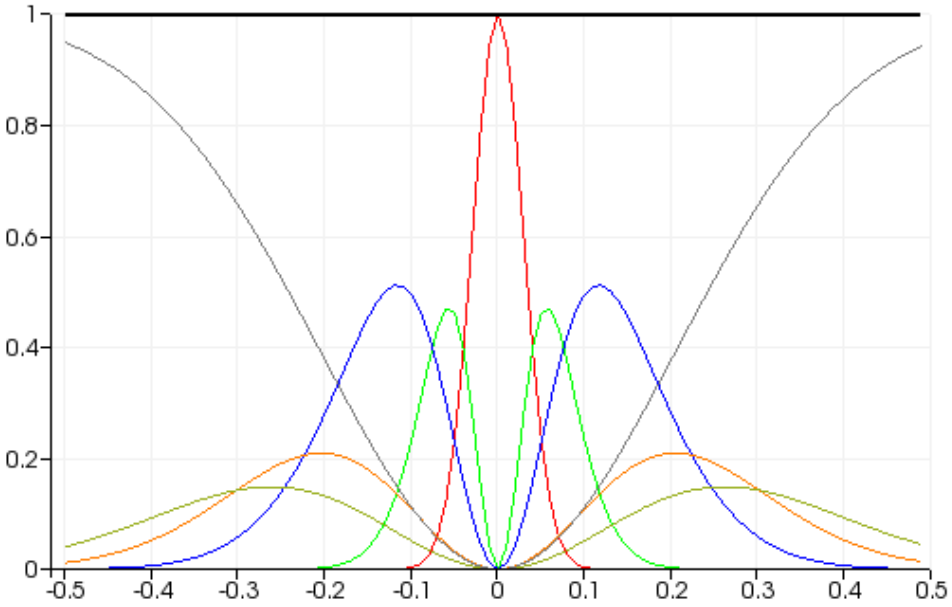

Abbildung 2.15: Vollständige Zerlegung des Spektralraums.

Aus Voraussetzung 2 folgt unmittelbar Voraussetzung 1: Um Ringing Artefakte zu vermeiden, müssen Bandfilter einen gewissen *'Roll off Value'* , also eine Übergangsfunktion mit endlicher Steigung besitzen. Dies bedingt allerdings auch, dass sich Bänder überlappen müssen um sich gemeinsam auf das ursprüngliche Signal zu ergänzen. Der Roll off Value gibt somit an, wie abgegrenzt Bandpassfilter auf in einem Spektralintervall definiert sind. Einen guten Kompromiss zwischen Definiertheit und Erfüllung von allen Voraussetzungen zu finden ist die schwierigste Aufgabe bei der Wahl der Filterfunktionen. Die Frage, die sich hier aufdrängt ist folgende: Wie genau müssen Filterkernel definiert werden, um die gewünschten Bandpassbilder zu erzeugen? Vuylsteke und Schoeters benutzen einen Ansatz, der auf der Zerlegung in eine Gauss-Laplace Pyramide beruht [14]. Dieser MUSICA Algorithmus ist wie folgt definiert: Ein Eingangsbild b wird tiefpassgefiltert (Gaussfilter) und auf die Hälfte heruntergesampled. Das entstandene Bild $b1'$ wird nur für die Berechnung der Differenz zum Originalbild hochskaliert, womit man das Differenzbild $d1$, die oberste Ebene der Pyramide, und das am feinsten aufgelöste Bandbild erhält. Danach wird auf das verkleinerte Bild $b1'$ wiederum ein Tiefpassfilter angewandt, welches $b2'$ liefert. $b2'$ wird hochskaliert und von $b1'$ abgezogen, welches die zweite Ebene $d2$ der Pyramide liefert. Im Sinne des Shannon' Abtastheorems ist es zulässig downgesamplete Bilder ohne Aliasing wieder upzusamplen, wenn für die Interpolation beim Upsampling und Downsampling dieselbe Methode verwendet wird.

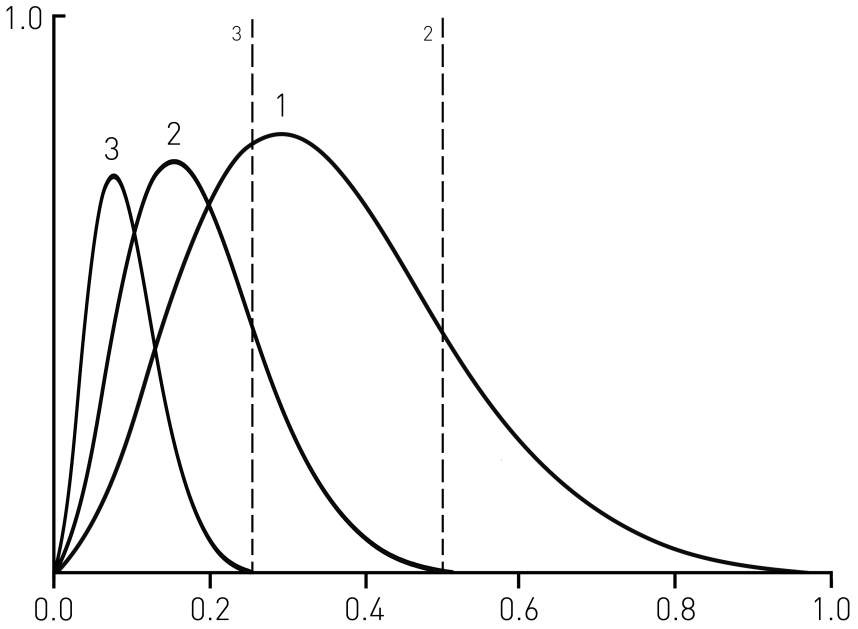

Abbildung 2.16: Querschnitt durch das Filterdesign von drei Bandpassfiltern einer Gauss-Laplace Pyramide. Durch die Verkleinerung des Bildes ergeben sich andere Ortsfrequenzen, auf die die Filter wirken.

Die Reduktion der Dimensionalität um einen Faktor wird damit begründet, dass die immer unschärfer werdenden Bandbilder ohnehin immer weniger feine Strukturmerkmale enthalten können, und somit eine hohe Bildauflösung nicht notwendig ist. Zusätzlich bewirkt dies, dass die Berechnung der Bandbilder schneller wird. In jedem Folgeschritt wird dieser Vorgang wiederholt, wobei Bandbilder immer geringerer Auflösung entstehen, solange bis ein Bild der Größe 1 x 1 entsteht. Die Synthese der Bandbilder zum Originalbild erfolgt durch hochskalieren der Bandbilder und pixelweiser Addition mit dem nächstgrößeren Bandbild. Die Rückkonstruktion des Originalbildes kann mit Gewichtungsfaktoren für die einzelnen Bänder vorgenommen werden, was einer individuellen Verstärkung bestimmter Wellenzahlen, und damit einer Kontrastanhebung der Bildstrukturen entspricht. Im diesem oben beschriebenen MUSICA Algorithmus basiert diese Verstärkung mittels einer Funktion die pixelwertekleinere Bandbilder stärker gewichtet, als wertegrößere. Der Verlauf der Funktion wird dabei von einem durch den Benutzer gewählten Faktor $(MUSI - EnhancementFactor)$ bestimmt. Abb. 2.16 zeigt einen Querschnitt durch die MTF von drei entstandenen Bandfilter. Man beachte, dass eine Verkleinerung des Bildes auch die Einschränkung des Bandfilters auf einen gewissen Wellenzahlbereich bewirkt.

In diesem Fall ergibt sich ein Differenzbild durch Subtraktion eines tiefpassgefilterten Bildes von der unmittelbar darüber liegenden (schwächer tiefpassgefilterten) Detailstufe. Ein Differenzbild ergibt sich somit aus:

$G1' = G * V1$
$G2' = G1' * V2$
$D1 = G1' - G2'$
$D1 = (G * V1 - G1' * V2)$

$G..Originalbild$

G1', G2' ... gefilterte Bilder
V1, V2... Filterfunktionen
D1... Erstes Differenzbild

2.5.5 Bildartefakte

Bildfehler, oder Bildartefakte stellen unerwünschte Störungen des Bildes dar, die ihren Ursprung in verschiedenen Ursachen haben. Bildfehler können sowohl während der Aufnahme in jedem Glied eines Optischen Systems auftreten, als auch durch falsche Bildverarbeitung auftreten. Man unterscheidet zwei Arten von Bildfehlern, systematische Fehler und statistische Fehler [5]. Systematische Fehler sind Fehler die erkennbar und quantifizierbar sind, und durch die Aufnahmemodalität zustande kommen. Solche Fehler können kompensiert werden, wenn man die Ursache kennt, und eine geeignete Bildtransformation existiert, die den Bildfehler wieder ausgleicht. Beispiele hierfür wären, ein Fehler in der Linse des Aufnahmesystems, fehlerhafte Positionierung des Aufnahmesystems oder ein fehlerhafter Bilddetektor. Systematische Fehler neigen dazu sich fortzupflanzen, es ist daher nicht immer einfach herauszufinden an welcher Stelle der Fehler auftritt. Systematische Fehler zu finden und auszubessern ist Gegenstand der Bildrestauration. Statistische Fehler gehorchen einer statistischen Verteilung deren Auftreten nicht vorhersagbar ist. Statistische Bildfehler beziehen sich immer auf das gesamte Bild, und sind in allen Frequenzen anzutreffen. Um statistische Fehler zu beeinflussen werden meist Methoden der Bildverarbeitung angewandt. Neben statistischen und systematischen Bildfehlern gibt es in der medizinischen Bildverar-

beitung noch Fehler die ihren Ursprung im aufzunehmenden Objekt haben , oder durch eine physikalische Besonderheit zustande kommen. Im Folgenden werden nur einige Beispiele gegeben.

Bewegungsartefakte (Motion Artefacts)

Bewegungsartefakte kommen durch Patientenbewegungen zustande, die das Bild verzerren lassen und treten dann auf, wenn die Scanzeiten des Aufnahmesystems länger dauern als die Bewegung. Beispiele wären Bewegungen des Herzens, und der Gefäße, das Senken des Brustkorbes oder das Bewegen einer Extremität. Da einige dieser Bewegungen unwillkürlich passieren sind Bewegungsartefakte ein häufig anzutreffendes Problem in der medizinischen Bildverarbeitung. Bewegunsgartefakte lassen sich vermeiden durch:

1. Kurze Scanzeiten.
2. Fixation, soweit möglich.
3. Getriggerte Aufnahme. Aufnahme erfolgt in Koordination mit der Entschlaffung des bewegungsauslösenden Muskels.(z.B: Herz)
4. Aufnahme eines Bildes zu verschiedenen Zeitpunkten, und anschließende Mittelung.

Gibbsches Ringing

Gibbsches Ringing entsteht wenn eines Signals mit Untstetigkeitsstellen fouriertransformiert wird. In unmittelbarer Nähe der Sprungstelle kommt es zu Über und Unterschwingungen gegen, die die Partialsumme der Fouriertransformierten über die eigentliche Funktion erhöhen oder erniedrigen, und das Signal mit künstlichen hochfrequenten Komponenten belasten. Wird zum Beispiel ein solches Filterkernel auf ein Bild angewandt macht sich dies durch Wellenbewegungen entlang von Kanten bemerkbar (Abb. 2.17).

Rauschen

Rauschen bezeichnet das statistisch verteilte Auftreten von Pixelstörungen und kann mehrere Ursachen haben. Von Quantenrauschen spricht man, wenn aufgrund der Poissonverteilung der Photonen nicht alle Bilddetekoren mit gleichmässiger Intensiät bestrahlt werden.

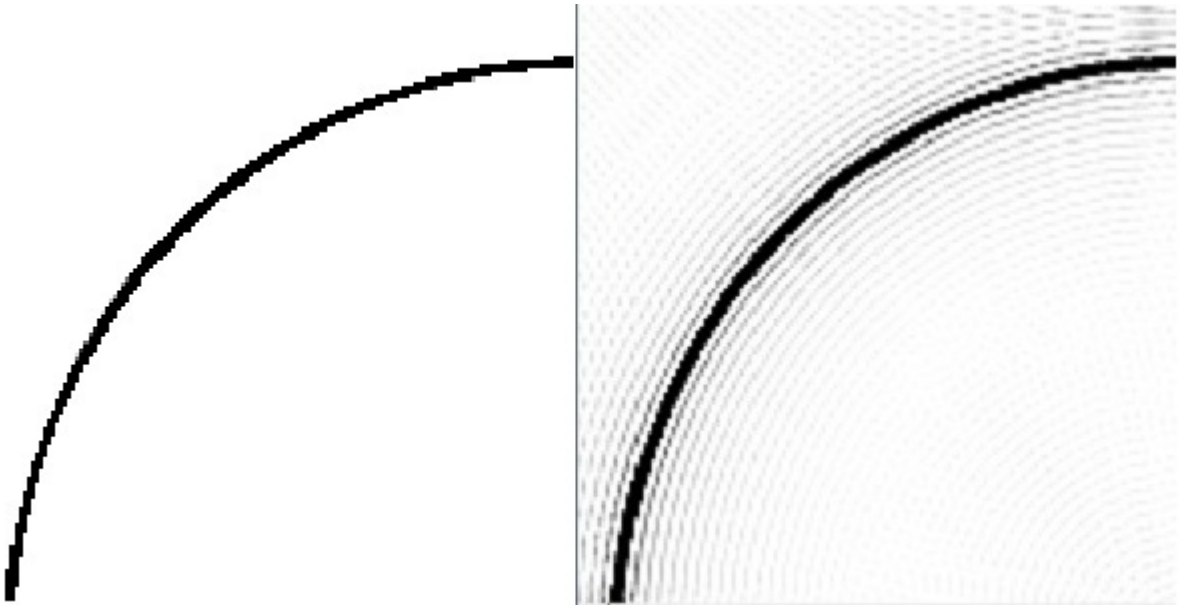

Abbildung 2.17: Gibbsches Ringing (l): original, (r): Wellenbewegungen durch Verwendung eines ungeeigneten Filters

Quantenrauschen ist im Allgemeinen durch eine höhere Photonenanzahl (Dosis) zu vermeiden. Von thermischen Rauschen spricht man, wenn bei digitalen Systemen durch den Betrieb des CCD Chips thermisch angeregte Elektronen für ein künstliches Signal sorgen und somit einen Helligkeitspunkt erzeugen. Thermisches Rauschen tritt bei allen digitalen Sensoren auf, und fällt bei besonders niedrigen Photonenintensitäten stärker ins Gewicht. Auf analogen Filmfolien entsteht Rauschen durch die Verteilung der Silberhalogenidkristalle, und den Inhomogenitäten in der Verstärkerfolie.

Aliasing Effekt

Der Aliasing Effekt entsteht durch eine Verletzung des Shannonschen Abtasttheorems, indem das Eingangssignal mit einer Abtastfrequenz kleiner der Nyquistfrequenz abgetastet wird. Wie in Absch.2.3 oben besprochen führt eine zu geringe Samplingrate dazu dass die Perioden einer abgetasteten Funktion im Frequenzraum näher zusammenrücken und miteinander interferieren. Die Folge ist, dass so eine gesampelte Funktion Frequenzen oberhalb der eigentlichen Nyquistfrequenz nicht mehr beinhaltet, und stattdessen die Frequenzen der Folgeperiode dargestellt werden. Das Ergebnis sind Interferenzerscheinungen des eigentlichen Signals mit der Abtastrate des optischen Systems. Aliasing tritt grundsätzlich immer beim Abtastvorgang auf, da die Impulsantwort der Rechteck Abtastfunktion unendliche Fre-

quenzkomponenten aufweist (vgl Box Filter) [9]. Deren Amplitude nimmt jedoch zunehmend ab, sodass der Aliasing Effekt der unendlich hohen Frequenzkomponenten bei korrekt gewählter Abtastrate vernachlässigbar ist.

Kapitel 3

Praktische Implementation

In diesem Kapitel wird auf die praktische Implementation des MFP Algorithmus in den zwei Frameworks AVW und ITK zunächst im Generellen eingegangen. Die grundsätzliche Arbeitsweise wird erklärt, und Besonderheiten und Unterschiede werden herausgearbeitet. Nach einer Kurzeinführung in die verwendeten Frameworks, werden alle 3 Programme Osirix, Analyze 10.0 und Slicer 3D seperat vorgestellt. Es werden ihre Stärken und Schwächen abgewogen und Unterschiede in ihrer Arbeitsweise, sowie in der individuellen Implementation des MFP Plugins aufgezeigt. Anhand der Vorgehensweise bei der programmierung des MFP Plugins wird herausgearbeitet, wie einfach oder schwer es für die 3 Programme ist, Erweiterungsplugins zu schreiben. Es wird darauf eingegangen was es für Vorrausetzungen für das Schreiben von Erweiterungen gibt, und was dabei speziell für den Entwickler zu beachten ist. Die jeweils erzielten Resultate werden danach durch Laufzeitberechnungen verglichen und bewertet.

3.1 Verwendete Frameworks

Im Folgenden wird eine kleine Übersicht über die verwendeten Frameworks und Bibliotheken gegeben, die bei der Implementierung des MFP Plugins verwendet wurden. Fast alle Frameworks sind unter einer Opensource Lizenz und für alle gängigen Betriebssysteme frei verfügbar.

3.1.1 ITK

ITK ist eine Opensource Software Library, geschrieben in C++, die für Bildmanipulation, Bildregistration, und Segmentation eingesetzt werden kann. ITK kann auch mit anderen Programmiersprachen, wie TCL, Phyton und Java mittels Wrapping Klassen verwendet werden, was der Bibliothek eine breite Einsetzbarkeit verleiht. Da die Bibliothek unter der Opensource Lizenz angeboten wird, können ITK Klassen beliebig für individuelle Bedürfnisse verwendet und erweitert werden. ITK ist plattformunabhängig einsetzbar, daher existieren für ITK keine prekompilierten Dateien; ITK muss selbstständig auf dem eigenen Computer kompiliert werden.

Zur Zeit der Erstellung dieser Arbeit konnte ITK unter http://www.itk.org/ heruntergeladen werden.

Itk Daten Formate

ITK kann eine Vielzahl von Bildformaten lesen und schreiben:

- Dicom
- GDCM
- Stimulate
- PNG
- VTK
- Gipl
- VOL

- BMP
- TIFF
- Raw
- Analyze
- Ge4x
- Jpeg
- nrrd
- SiemensVision

Andere Dateiformate können eingelesen werden, wenn eine entsprechende itk Importerklasse verwendet wird.

Besonderheiten von ITK

Generische Programmierung Die große Stärke von ITK liegt im Konzept der generischen Programmierung. In generischer Programmierung werden statt konkreten Klassen von Objekten 'Template Klassen' von Objekten verwendet. Template Klassen beinhalten quasi eine 'Bauanleitung' in der steht welche Merkmale Objekte, die zu dieser Klasse gehören, haben müssen. Über die genaue Implementierung wird allerdings nichts im Template ausgesagt. So kann z.B ein Bild native Datentypen wie float, int, char usw für Pixelwerte annehmen, es ist allerdings genauso möglich einen eigenen Pixeltyp zu definieren (z.B: komplexe Bildpunkte im Falle einer Fourieranalyse). Template Klassen sind somit sehr flexibel und erlauben es ein und derselben ITK Klasse vielfältig eingesetzt zu werden.

ITK Data Processing Pipeline Verarbeitungprozesse in ITK sind am besten anhand eines Pipelinemodells vorstellbar. Im unten angeführten Beispiel wird eine Verabeitunsgpipeline zwischen $Filter_{A,B,C}$ festgelegt, wobei jeweils der Output des vorhergehenden Filters Input für den nachfolgenden ist. In ITK ist die Abarbeitung einer Prozesspipeline rekursiv definiert,d.h wird von hinten nach vorne abgearbeitet. Wenn im Beispiel unten $Filter_C$ einen Output produziert, berechnet er zuerst den Output von $Filter_B$, welcher wiederum den Output von $Flter_A$ benötigt.

$$Filter_A \longrightarrow Filter_B \longrightarrow Filter_C \longrightarrow GetOutput[Filter_C]$$

Diese Art der Abarbeitung benötigt ITK, da zu Beginn der Ausführung für jedes Glied der Kette berechnet wird:

1. Wieviel Speicherplatz der Filter benötigt, und mit welchem Datentyp er arbeitet.
2. Wann der Speicherplatz wieder freigegeben werden kann.
3. Im Falle von Multithreading können die Verarbeitunsgschritte besser auf die Prozesse optimiert werden.

In Summe bringt dies einen Geschwindigkeitsvorteil.

3.1.2 VTK

VTK ist ebenso wie ITK eine freie Bibliothek, geschrieben in C++, die auf die visuelle Darstellung von Daten spezialisiert ist. VTK umfasst bereits über 1000.000 Zeilen Code, und Schnittstellen zu anderen Programmiersprachen, wie etwa Phyton, oder Java und ist auf Windows, Linux und Mac Systemen gleichermaßen verfügbar. VTK wird von der Firma Insight vertrieben, die Schulungen und Kurse für VTK anbietet. VTK besitzt eine Vielzahl von Visualisierungsalgorithmen, z.B: für Skalare, Vektoren, Tensoren,Texturen, und 3D Darstellung, sowie erweiterte Modellierungstechniken wie: Implicit Modeling, Polygon Reduction, Mesh Smoothing, Cutting, Contouring, und Delaunay Triangulation. VTK besitzt einige der Konzepte von ITK. So haben sie beide dasselbe Speichermanagement der Smart Pointer, sowie die Staffelung von Verarbeitungsprozessen in Form von Pipelines. Anders als ITK besitzt VTK keine generische Programmierung: Jeder ITK Filter benötigt einen bestimmten Inputtyp. VTK und ITK besitzen Schnittstellen zueinander, was es ungemein erleichtert, Daten in ITK zu berechnen und von VTK darstellen zu lassen. Der einzige Unterschied ist, dass das Koordinatensystem von Bildern bei VTK seinen Ursprung im linken unteren Eck hat, bei ITK jedoch im linken oberen Eck. Im Rahmen dieser Arbeit wurde VTK zur Darstellung der Bilder und der Gauss Filtermasken verwendet. Es wurde eine Klasse geschrieben, die von der vtk Klasse vtkContextView abgeleitet und verändert wurde, sodass ihr ein vtkRenderWindow übergeben werden kann, auf dem sie zeichnet. Die Bilddarstellung selbst erfolgt mittels der Klasse vtkRenderWindow2, welche ein Fenster zum rendern, sowie alle Funktionen zum Einstellen und Verändern des Window Levels und der Breite enthält. Für den Support zwischen VTK und ITK Pipeline musste zusätzlich ein vtkImageFlipFilter verwendet werden, um das Koordinatensystem zwischen ITK und VTK abzugleichen.

Zur Zeit der Erstellung dieser Arbeit konnte VTK unter http://www.vtk.org/ heruntergeladen werden.

3.1.3 KWWidgets

KWWidgets ist eine platformunabhängige, freie Ansammlung von C++ Bibliotheken, die grafischen Benutzeroberflächen mithilfe der Programmiersprache TCL erstellt. TCL selbst dient dabei dem Programmierer nicht als Programmiersprache: Jeglicher Sourcecode ist komplett in C++ verfassbar. KWWigets wird genauso wie VTK von Insight vertrieben, und besitzt viele Schnittstellen zu VTK Objekten, wird allerdings nicht mehr weiterentwickelt.

Zur Zeit der Erstellung dieser Arbeit konnte KWWidgets unter http://www.kwwidgets.org/ heruntergeladen werden.

3.1.4 AVW

AVW ist eine von der Firma Mayo Clinic kommerziell vertriebendes, lizenzpflichtige Library zur Bearbeitung, Segmentierung Registrierung und Darstellung von einer Vielzahl von Datenformaten. Die AVW Library wird intensiv von dem ebenfalls von Mayoclinic vertriebenen Produkt Analyze genutzt,von der Analyze fast alle seine Funktionalitäten bezieht. Die AVW Library kann in verschiedene Unterbereiche aufgeteilt werden [10]:

- Transform
  Verschiedne Arten von Transformationen: räumliche, densiometrische, und mathematische. Diese Funktionen kommen z.B bei der Registration oder bei der multiplanaren Rekonstruktion zum Einsatz.
- Process
  Bildfilterfunktionen im Orts und Frequenzraum.
- Segement Funktionen zur Segmentierung: Treshholding, Region Growing, Multy Spectral Decomposition, Watershed Segmentierung. Manche der Segmentationsalgorithmen sind halbautomatisch.
- Analysis: Funktionen zur statistischen Auswertung von Daten.

- Visualize: Funktionen zur 2D & 3D Visualisierung von Datensätzen / Object / Surface Maps
- Tiling / Modeling: Erstellt AVW Object & Surface Maps, die aus Segmentationsschritten gewonnen wurden, und als Basis für andere Funktionen (Klassifikation / Visualisierung) dienen.

AVW Daten Formate

AVW kann eine Vielzahl von Daten laden und speichern, sowie unbekannt Datenformate lesen, wenn Bildaten unkomprimiert und in in kontinuierlicher Reihenfolge gespeichert sind. (Gemeint damit ist ein zusammenhängender Bytestream der Bilddaten). Im Folgenden sind alle lesbaren Datentypen aufgelistet.

Analyze spezifische Daten

- AnalyzeAVW: Standard Analyze Bildformat.
- ANALYZEIMG : Altes Analyze Bildformat (7.5).
- VolumeFile: Stack von 2D Bildern.
- OBJMAP: Analyze object map Format.
- ANALYZESCRN: Altes Analyze Format.
- UNKNOWN(raw): Erlaubt es rohe Bilddaten zu lesen, für die sonst kein Support besteht.

Radiologische Standard Bilddaten

- DICOM: Standard Medizin-Bildformat.
- ACR/NEMA: Vorgänger von DICOM.
- PAPYRUS2: Papyrus 2D File.
- PAPYRUS3: Papyrus 3D File.
- NIfTI: Neurologisches Bildformat (NIH).

Hersteller spezifische Bilddaten GE9800,GEADVANCE ,GEADVANTAGE ,GESIGNA ,GESTARCAM ,INTERFILE ,PICKER MRI , SIEMENSCT ,SIEMENSMAGVIS ,CTI ,7CTI ,IMATRON ,SMIS ,BRUKER MR I,VARIAN FDF BIORAD.

Standard Bilddateien BMP,GIF ,JPG ,PBM ,PGM ,PPM ,PIC ,PNG ,PS ,SGI ,SUNRASTER ,TARGA ,TIFF ,XBM.

Video QUICKTIME ,YUV , AVI, MPEG1.

3.2 Implementation des MFP Algorithmus

In Absch.2.5.4 wurde eine Methode von Vuylsteke & Schoeters beschrieben, die auf der Zerlegung der Bilddaten in eine Gauss-Laplace Pyramide beruht [14]. Die im Laufe dieser Arbeit angewandte Methode des MFP basiert nicht auf einer Gauss-Laplace-Pyramide, aber ebenso auf der Anwendung des sogenannten Difference of Gaussians Prinzips: Bandbilder ergeben sich durch Subtraktion zweier gaussgefilterter Bilder. Allerdings wird dabei immer das Originalbild zur Tiefpassfilterung verwendet. In diesem Fall ergibt sich für die Berechnung eines Differenzbildes:

$$G1' = G * V1$$
$$G2' = G * V2$$
$$D1 = G2' - G1'$$
$$D1 = G * (V2 - V1)$$

G1', G2'.... Gaussgefiltertes Bild
G ... Originalbild
V1,V2 ... Gaussfilter
D1 ... Differenzbild

Die Subtraktion der Bandpässe ist also dasselbe, wie die Subtraktion der Filterkernel und anschließender Multiplikation mit dem Originalbild. In dieser Methode werden die entstandenen Bilder auch nicht downgesampled. Dies hat den Nachteil, daß Berechnungen der Bandbilder viel länger dauern. Vuylsteke und Schoeters verwenden für ihre Bandpässe immer Gaussfilter gleicher Größe. Die entstandenen Bandbilder nehmen somit immer gleichviel Platz im Fourierraum ein, und es ist daher nicht automatisch gegeben, dass einer der Bandpässe sein Maximum an einer gewünschten Position im Fourierraum besitzt. In der hier angewandten Methode werden Filterkernel mit sukzessive breiter werdender Halbwertsbreite eingesetzt. Um eine vollständige Zerlegung zu erzielen müssen Bandbilder immer durch Subtraktion eines größeren Filterkernels von einem kleineren generiert werden. Dadurch ist das Filterdesign soweit wählbar, dass die Position des Maximums eines Bandes mit Einschränkungen frei bestimmbar wird. Da sich ein Differenzband immer aus der Subtraktion eines kleineren Gaussfilters von einem größeren ergibt, wird durch die Variation des Standardabweichung eines Bandes auch gleichzeitig das Aussehen des nachfolgenden (hin zu höheren Ortsfrequenzen) beeinflusst. Die Position des Maximums eines Bandes kann somit

durch eine Kombination der Verbreiterung/Verschmälerung des Bandes selbst, und durch Verbreiterung/Verschmälerung des Bandes unmittelbar davor (zu niedrigeren Frequenzen hin) bewerkstelligt werden. Abb. 3.1 macht dies deutlich: In diesen Beispielen wurden 4 Bänder mit einer gewissen Breite erzeugt. Ziel war es Band 2 auf einem Linienpaar/mm Wert von 0,1 einzustellen.Im ersten Fall wurde das erste Band (rot) verändert, um das zweite Band (grün) auf einen Linienpaar/mm Wert von 0,1 zu verschieben. Im zweiten Fall wurde Band 1 von seinen vorhergehenden Einstellungen nur wenig verschmälert, wodurch sich das Maximum von Band 2 nach vorne (< 0,1 Linienpaare/mm) verschob. Um das Maximum von Band 2 wieder auf 0,1 Linienpaare / mm zu verschieben, wurde die Standardabweichung von Band 2 stark vergrößert. Dies führt aber zusätzlich noch zu einer Veränderung von Band 3 (blau), da dieses durch die Verbreiterung von Band 2 schmäler werden muss. Band 4 (orange) bleibt in beiden Fällen gleich, da es sich erst ändern würde, wenn sich die Standardabweichung von Band 3 änderte. Die Form eines Bandes variiert also mit der Position seines Maximums, welches sich aus der Kombination der Standardabweichung für dieses Bandes, und der des Bandes unmittelbar davor (zu niedrigeren Frequenzen hin) ergibt. Dies macht es für die hier eingesetzte Variante des MFP unmöglich, Bänder seperat voneinander auf ein gewünschtes Maximum einzustellen. Ein Wunschmaximum bedingt immer, dass sich insgesamt bis zu 3 zusätzliche Bänder mitändern. Ausserdem kommt es zu einer starken Überlappung der Bandbilder durch die Subtraktion der Gausskurven. Da der Gaussfilter nur in einem Punkt 0 wird (im Ursprung) enthalten alle Bänder zusätzlich noch Frequenzen der weitere innenliegenden Bänder. Dies ist nicht unbedingt sinnvoll, weil es eine klare Seperation von bestimmten Ortsfrequenzen erschwert. Die entsprechenden Teilbandbilder besitzen somit zu großen Anteilen gleiche Ortsfrequenzen, und damit Strukturmerkmale.

3.2.1 Implementation in ITK

Für die die Implementation des MFP Algorithmus in ITK wurden einige ITK Klassen verwendet, andere wurden selbst geschrieben:

- VnlFFTRealToComplexConjugateImageFilter:
  Dieser ITK Filter transformiert die Eingabedaten mittels FFT in den Frequenzraum.
- FFTShiftFilter:
  Dieser Filter wurde selbst geschrieben, und zentriert die Daten in den Ursprung.

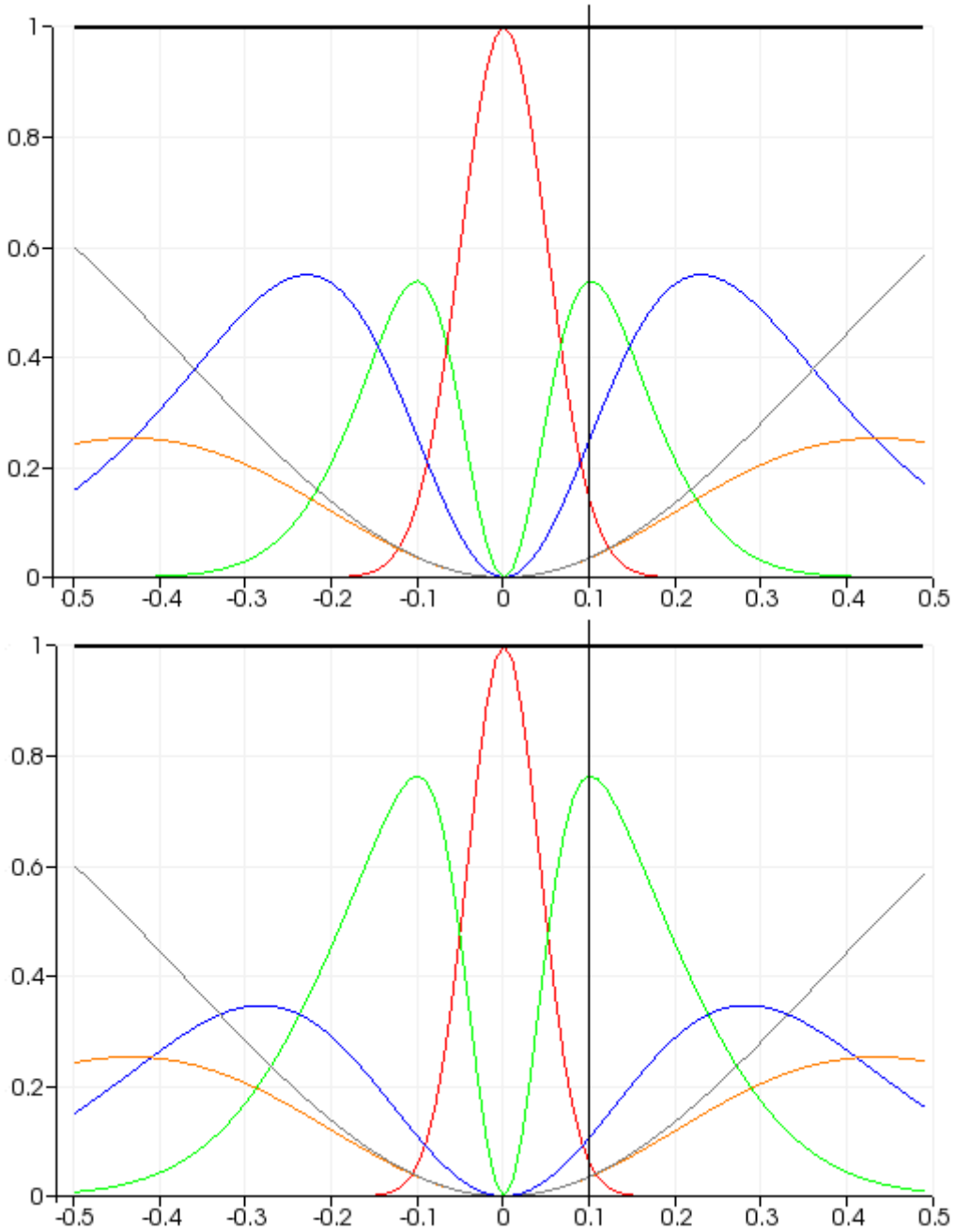

Abbildung 3.1: In beiden Fällen befindet sich das Maximum des zweite Bandes (grün) auf 0,1 Linienpaaren/mm, allerdings ändert sich dabei die Form und/oder das Maximum der anderen Bänder.

- VnlFFTComplexConjugateToRealImageFilter:
  Originaler ITK Filter, der die inverse Fouriertransformation durchführt.
- GaussMaksFilter:
  Selbstgeschriebener Filter, der ein Eingangsbild mit einer Gaussmaske multipliziert.

In Abb. 3.2 ist der genaue Ablauf der ITK Pipeline zu erkennen. Die Voraussetzungen für die ITK Piepline ist ein dreidimensionales itk::Image mit Datentyp float. Daher müssen die Eingangsdaten zunächst in ein float image konvertiert werden, falls sie noch nicht im float Format vorliegen. Der Datentyp float wird benötigt, da die Multiplikation des FFT Bildes mit einer Gaussfilter Maske Fließkommazahlen voraussetzt. Der erste Schritt ist eine Überprüfung auf FFT Konformität der Daten. Die in ITK implementierte FFT kann dabei Daten mit 2er oder 3er Potenzen verarbeiten. Zusätzlich wird das Bild quadratisch angepasst , um eine anisotrope Verteilung der Frequenzen im Fourierraum zu garantieren, sowie das Bild in ein Bild doppelter Größe mittels Padding eingebettet, um Wrap Around Artefakte zu vermeiden. Die Transformation in den Frequenzraum geschieht mittels VnlFFTRealToComplexConjugateImageFilter. Da die Periode nach diesem Schritt aufgrund der Implementation der DFT so aufgeteilt ist, dass am Beginn und am Ende der Daten jeweils eine Halbperiode zu finden ist, wird die Bildmatrix mittels des FFTShiftFilters so verändert, dass der Ursprung der Periode in den Mittelpunkt der Bildmatrix verlegt wird. Danach wird die nun zentrierte Fouriermatrix mittels eines zentrierten Gaussfilters multipliziert. Die wird einmal für das erste Band mit der größten Standardabweichung (große PSF, kleine MTF) gemacht. Für die nachfolgenden Bänder wurde der GaussMaskFilter so konzipiert, dass er bei Eingabe von zwei Standardabweichungen das entsprechende Subtraktionsbild (kleinere Standardabweichung - größere Standardabweichung / schärferes Bild - unschärferes Bild) erstellt. Die Standardabweichungen die der Benutzer hier eingibt entsprechen den Standardabweichungen für den Gaussfilter im Fourierraum, da eine Gausskurve durch eine FFT nur ihre Form ändert nach wie vor aber eine Gausskurve bleibt (vgl. Abschn. 2.5.3). Für die Rücktransformation des entstandenen Bildes wird zunächst noch einmal ein FFT Shift angewandt, bevor die Daten mit einem VnlFFTComplexConjugateToRealImageFilter wieder in den Bildraum zurück transformiert werden. Die Realdaten werden daraufhin mit den entsprechenden Faktoren gewichtet, und in ein lokales Array geschrieben. Dieser Vorgang wird 6 Mal, für die 6 Bänder, die dem Benutzer zur Verfügung stehen, mit den von ihm gewählen Standardeinstellungen vorgenommen. Am Ende jeder dieser Schritte werden die Teilbilder aufsummiert. Beim 7 ten Mal wird statt des Gaussfilters als Tiefpass ein inverser Gaussfilter, der als Hochpass fungiert, mit der größten eingestellten Standardabweichung verwendet. Der Hochpass ist als Gütekriterium

zu verstehen, das angibt wieviel Restinformation noch im letzten Bildabschnitt steckt. Der Hochpassfilter wird bei zunehmenderer Breite der Gausskurven immer weniger Bildinformation, und immer mehr hochfrequentes Rauschen beinhalten, wohingegen er bei schmalen Gausskurven einen Größtteil der Bildinformation erhält. Zuguterletzt muss eine Histogrammanpassung vorgenommen werden: Durch das gewichtete Aufsummieren aller Teilbilder ändert sich das Histogramm des Ergebnisbildes, abhängig von der Stärke der Gewichtung, in Richtung des am stärksten gewichteten Teilbildes. Dies führt bei einer Ansicht mit gleichen Grauwerteinstellungen (gleichem Windowlevel und Breite) zu einer verschlechterten, kontrastarmen Ansicht des Bildes. Aus diesem Grund wurde ein itkHistogramMatchFilter angewandt, der das Ergebnisbildhistogramm auf das Histogramm des Originalbildes angleicht. Originalbild und Histogramm besitzen somit eine annähernd gleiche Verteilung der Helligkeitswerte. Nachdem dieser Verarbeitungsschritt beendet ist, werden die Daten wieder in das lokale Array geschrieben. Wenn alle Faktoren mit 1 gewichtet wurden, ist das Ergebnisbild und das Ausgangsbild identisch. Jeder der einzelnen Verarbeitungsschritte bekommt den ganzen Datensatz des vorigen Schrittes als Input, und erzeugt ein neues Bild als Output. Dies ist eine Standardeinstellung bei der Verwendung eines ITK Filters und kann bei großen Datensätzen und 32 Bit Systemen zu Fehlern führen. Da es sich um mehrere Verarbeitunsgschritte handelt, die Daten alle in float gelesen/konvertiert werden, und die komplexen Bilder aus zwei float (IM, RE) pro Pixel bestehen, kann hier eine enorme Datenmenge zusammenkommen, die den Arbeitsspeicher von 32 Bit Systemen schnell überladen kann. Die ITK Pipeline arbeitet zudem mit Smartpointern, deren Funktionsweise einem Garbage Collector ähnelt. Das bedeutet, dass Speicher nicht direkt vom Benutzer freigegeben werden kann, sondern der Zeitpunkt der Zerstörung eines Objektes und damit die Freigabe seines Speichers vom System selbst gewählt wird. Durch eine Verändeung der Pipeline könnte man dieses Problem in den Griff bekommen: Itk bietet die Möglichkeit der sogenannten 'inplace' Filter. Das bedeutet dass Filter, nicht wie standardmässig, einene neuen Datensatz erzeugen, sondern ihre Berechnungen auf dem selben Datensatz durchführen. Konkret würde dies bedeuten, dass alle Rechenoperationen die auf komplexen Spektraldaten beruhen vereinbar sind, d.h. dass der erste FFTShift, die Multiplikation mit dem Gauss Filter, sowie der zweite FFTShift auf denselben Daten durchgeführt werden könnte. Eine weitere Möglichkeit wäre die Verwendung von Streaming Filtern. Jeder Filter würde in diesem Fall immer nur einen Teildatensatz übergeben werden. Dies führt dazu dass es nicht so stark ins Gewicht fällt wenn der Speicherplatz eines Filter der Pipeline nicht sofort wieder freigegeben würde. Zusätzlich könnte man durch parallele Verarbeitung, mit der Verwendung einer Grafikkarte die Berrechnungszeit weiter senken.

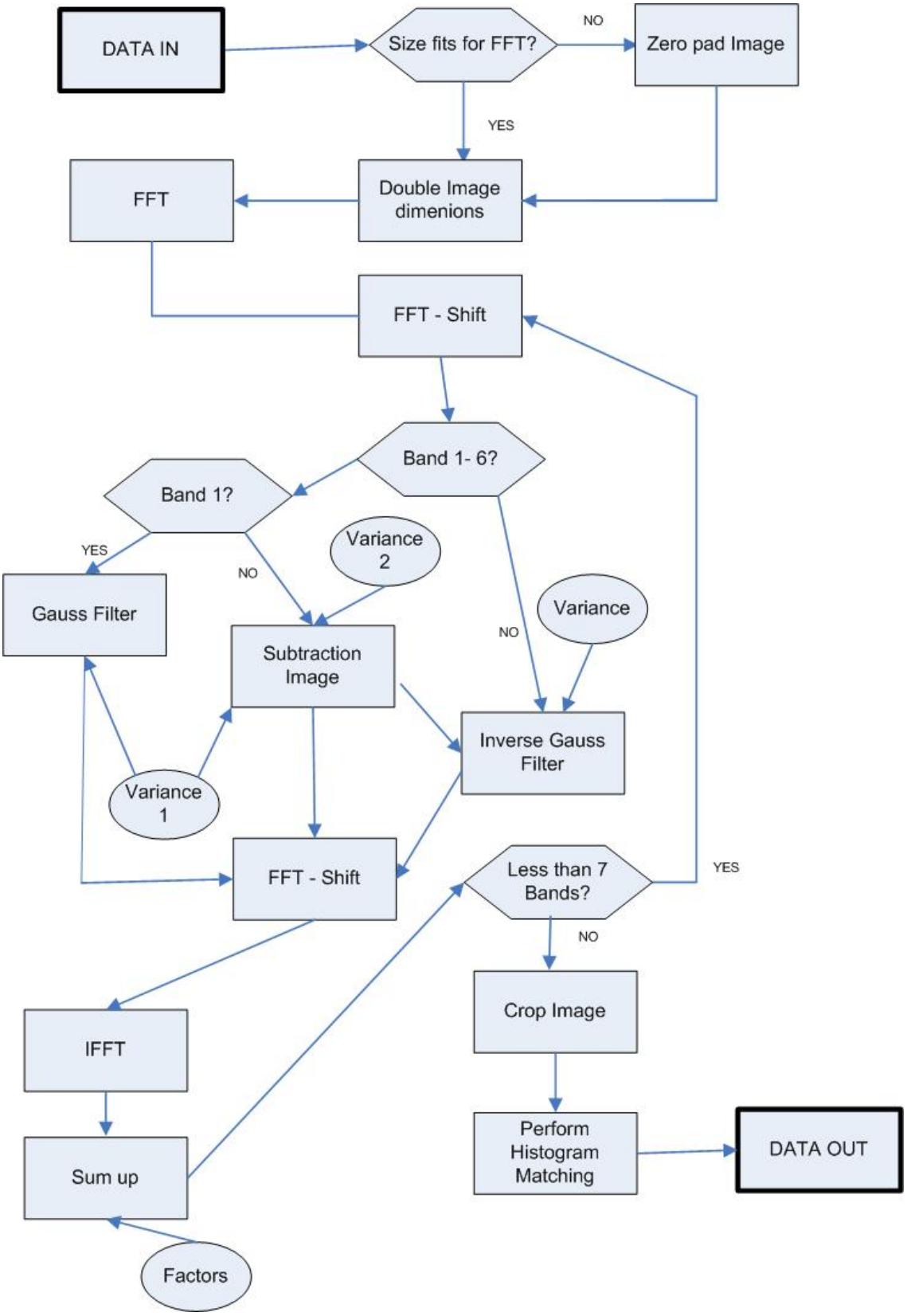

Abbildung 3.2: Der schematische Abblauf des MFP Algorithmus für ITK.

3.2.2 Implementation in AVW

Im Folgenden wird eine Übersicht gegeben, wie der MFP Algorithmus in AVW implementiert wurde. Da die AVW Library nur als bereits vorkompilierte Version verfügbar ist, ist es schwer einen genauen Einblick darin zu erhalten was genau ein Filter tut. So ist es z.B: bei AVW unerheblich, welche Dimensionen ein Datensatz hat um eine FFT auf ihn anzuwenden. AVW vergrößert Datensätze intern auf eine FFT kompatible Dimension , wendet die FFT an, und verkleinert den Datensatz wieder, ohne dass der Benutzer davon etwas mitbekommt. Weiters benötigt das FFT transformierte Bild auch keinen FFT Shift, der die Wellenkomponenten mit hoher Intensität in die Mitte verschiebt. Im Prinzip ist aber der Ablauf des Algorithmus genauso nachzuvollziehen wie bei der ITK Pipeline, bis auf einen kleinen Unterschied:Die AVW Pipeline ist aufgrund des Unterschieds zwischen AVW 2D und 3D Datentypen nicht auf 3D Stapel anwendbar. (Im Gegensatz zur ITK Pipeline). Der Ablauf ist wie folgt: Ein Eingangsdatensatz wird mittels FFT in den Fourierraum transformiert. Eine Besonderheit von AVW ist, dass hier eine Konversion zwischen AVWImageFile und AVWImage gemacht werden muss. AVWImageFile stellt quasi nur eine Importerklasse dar, AVWImage hingegen ist ein AVW Objekttyp mit dem alle Berechnungen durchgeführt werden. Danach definiert der Benutzer Standardabweichungen für Gaussfilter im Fourierraum, mit denen das Eingangsbild dann multipliziert wird. Das erste Bild stellt dabei das am gröbsten gaussgefilterte Bild dar. Bei den übrigen Bildern handelt es sich um Differenzbilder. Die Differenzbilder werden durch Subtraktion eines gaussgefilterten Bildes größerer Standardabweichung (große PSF, kleine MTF) von einem mit kleinerer Standardabweichung (kleine PSF, große MTF) erstellt. Dies wird 6 mal gemacht, für 6 verschiedene Bänder. Das letzte Band ist ein Hochpass, der das Restband zur Vervollständigung des Frequenzraumes darstellt. Da aus einem unbekannten Grund nur der Gausssche Tiefpassfilter in AVW implementiert wurde, nicht aber ein Gaussscher Hochpass, musste die Hochpassfiltermaske durch Subtraktion der Gaussfiltermaske von einer Einheitsmatrix erstellt werden. Abb. 3.3 gibt eine Übersicht über den Ablauf der AVW Pipeline.

3.2.3 Unterschiede AVW / ITK

Der MFP Algorithmus wurde mit den zwei Frameworks ITK und AVW implementiert, Tabelle 3.1 gibt eine Grobübersicht über die fundamentalen Unterschiede.

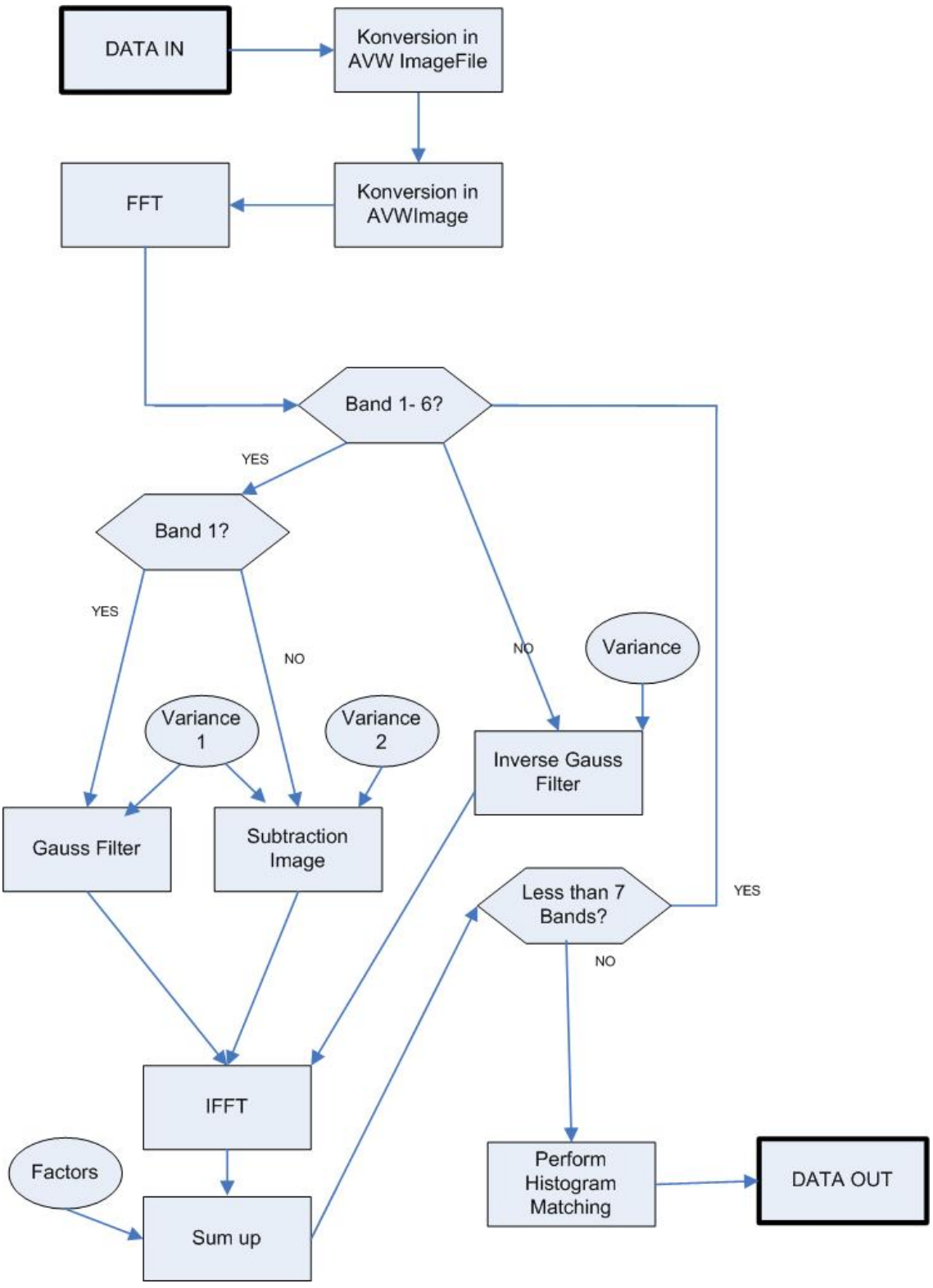

Abbildung 3.3: Der schematische Abblauf des MFP Algorithmus für AVW.

AVW	**ITK**
Unterschiedliche Datentypen für 2D / 3D	Ein Datentyp
Objekte müssen gelöscht werden	Smartpointer löschen selbstsändig (ähnl. Garbage Collector)
Unabhängige Abarbeitung (Individuelle Abfertigung von Programmschritten)	ITK Prozess Pipeline (optimiert Programmschritte)
Einsicht in Prozesse unklar	Open Source
Lizenzpflichtig	Frei verfügbar
Nicht erweiterbar	erweiterbar
Kann denselben Speicher eines Bildes öfter verwenden	Standardmässig wird immer ein neues Bild erstellt(Speicherbedarf!!)
Bildtypen müssen mit extra Libraries geladen werden	Keine zusätzlichen Libraries

Tabelle 3.1: Unterschiede zwischen AVW und ITK

3.2.4 Ergebnisse

Im Folgenden soll eine kurze Demonstration der Wirkungsweise des implementierten Multy Frequency Filters gegeben werden. Nimmt man z.B das Schulterröntgen (Abb. 3.4), so zeigt sich eine starke Überbelichtung im Bereich der Rippenbögen und des Schulterblattes. Durch geeignete Grauwertfensterung ist diese Überbelichtung zwar behebbar, gleichzeitig senkt dies aber den Kontrast in anderen Regionen soweit, dass Strukturen dort verschwinden. Mit Hilfe des MFP können die sich langsam ändernden Wellenzahlen (homogene Bereiche) abgeschwächt werden, sodass man einen besseren Kontrast erhält. Abb. 3.5 zeigt, dass die Gesamtintensität der Helligkeitswerte ungefähr gleich geblieben ist, der Kontrast jedoch angehoben wurde.

Ein weiteres Beispiel zeigt Abb. 3.6, die Röntgenaufnahme einer Wirbelsäule. Wie man sehen kann ist die Kontrastauflösung im Bereich der Wirbel sehr schlecht. Mithilfe des MFP wurden die Strukturen im cm Bereich (Größenördnung des Rückrades) abgeschwächt, sodass das bearbeitete Bild dort einen größeren Kontrastsunterschied aufweist. Jeder einzelnen Wirbel ist nun klarer zu erkennen, und sogar die Dornfortsätze können gut seperiert werden (Abb.3.7). Ein anderes Beispiel zeigt die Röntgenaufnahme eines Hundekopfes (Abb. 3.8). Hier wurden die niedrigen Frequenzen abgeschwächt, gleichzeitig aber die hohen

Abbildung 3.4: Röntgenabbildung der linken Schulter mit Überbelichtung [6]

Abbildung 3.5: Schulter aus Abb. 3.4 mit einer Abschwächung der niedrigen Frequenzen

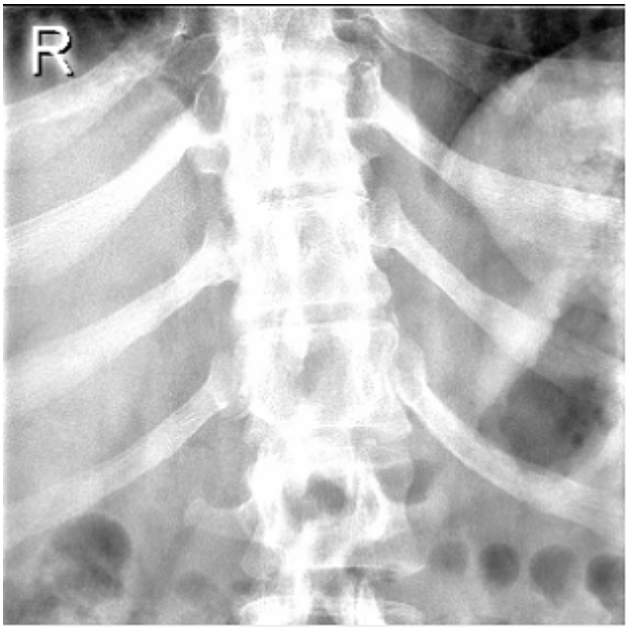

Abbildung 3.6: Wirbelsäulenröntgen mit niedriger Kontrastauflösung entlang des Rückrades

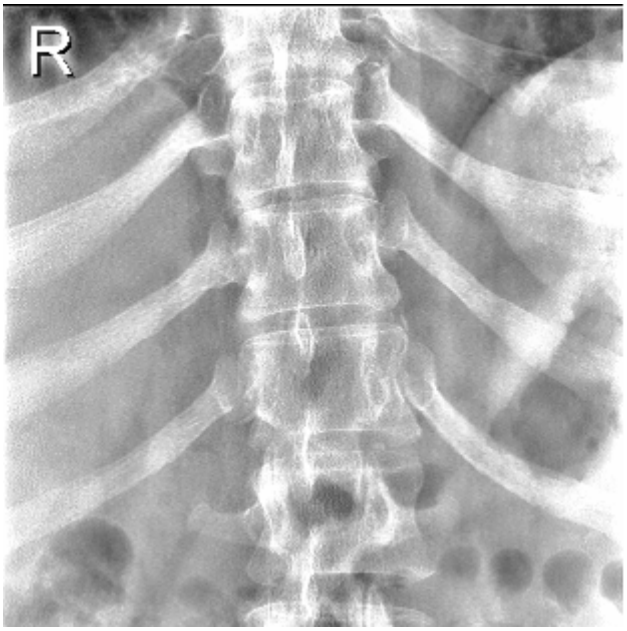

Abbildung 3.7: Wirbelsäule aus Abb.3.6 nach Anwendung eines MFP.

Abbildung 3.8: Hundekopf [4]

angehoben. Das Ergebnis ist ein kontrastreicheres, geschärftes Bild (Abb.3.9)

Abbildung 3.9: MFP bearbeiteter Hundekopf aus Abb. 3.8

3.3 Slicer 3D

3.3.1 Einleitung

Slicer3 [1] ist eine opensource Bildverarbeitungssoftware zur Visualisierung und Analyse von Bilddaten. Slicer3 bietet mehrere Segmentations & Registrierungsmethoden unterschiedlicher Bildmodalitäten, verschiedene Projektionsabbildungen der Bilddaten, sowie 3D & 4D Visualisierung an. Zusätzlich lässt sich die Funktionalität beliebig erweitern da es die Möglichkeit gibt Plugins zu schreiben die dann in die Software integriert werden. Die derzeitige Version von Slicer (3.6) besitzt gut 50 Plugins, unter anderem solche für statistische Bildanalyse, Diffusion Tensor Imaging (DTI), Functional Magnetic Resonance Imaging (fMRI) und Image-Guided Therapy. Slicer3 startete 1998 als Master Thesis Projekt zwischen dem Surgical Planning Laboratory am Brigham and Women's Hospital in Boston und dem MIT Artificial Intelligence Laboratory und hat in der Zwischenzeit eine große Community aus Entwicklern und Endbenutzern die ständig weiter wächst. Um so mehr da Slicer3 für Windows, Apple, Solaris und Linux Betriebssysteme entwickelt wurde, und 32 Bit bzw. 64 Bit Versionen der Software verfügbar sind. Das Projekt wird aus mehreren dem National Institute of Health (NIH) unterstellten Instituten betreut und finanziert, unter findet Unterstützung von mehreren amerikanischen Instituten, wie dem National Institute of Biomedical Imaging & Biomedical Engineering, dem National Cancer Institute, der National Science Foundation, sowie dem Amerikanischen Veteidigungsministerium. Slicer3 wurde unter dem Aspekt der freien Softwareverwend- und weiterverwendbarkeit entwickelt und darf vom Standpunkt der Sourcecode-Lizenz beliebig weiterverwendet oder weiterentwickelt werden. Slicer3 besitzt in Amerika keine FDA (Food & Drug Administration) Lizenz, ist also nur für klinische Forschung aber nicht für die Therapie zugelassen!

3.3.2 Installation

Für die Installation von Slicer3 bieten sich von Anfang an zwei Optionen:

1. Eine vorkompilierte Version.
2. Eine selbstkompilierte Version.

[1] Slicer3 - http://www.slicer.org/

Zum Zeitpunkt dieser Arbeit konnte Slicer3 über http://www.slicer.org/pages/Special:SlicerDownloads heruntergeladen werden.

Die vorkompilierte Version

Auf der Download Seite von Slicer3 findet sich die Möglichkeit die aktuelle Version von Slicer3 für das entsprechende Betriebssystem herunterzuladen. Zum Zeitpunkt dieser Arbeit waren für Windows nur 32 Bit Versionen, für Linux und Mac allerdings auch 64 Bit Versionen verfügbar. Bei der Versionsauswahl kann man zwischen der vorläufig aktuellsten Version (stable release) und einer sich in der Entwicklung befindlichen Version entscheiden. Letztere ist nur für Profis gedacht, und sollte nicht von Anfängern benutzt werden. Alle hier aufgelisteten Varianten sind bereits vorkompiliert, bei denen man nur mehr den für den Computer lesbaren Maschinencode herunterlädt, nicht aber den Sourcecode. Die Verwendbarkeit dieser Versionen ist leider sehr eingeschränkt:

1. Es lassen sich keine neuen Änderungen durchführen, da der Sourcecode fehlt.
2. Nicht alle eigens entwickelten Plugins lassen sich mit der vorkompilierten Version starten.

Wenn man sich für einen dieser Versionen entscheidet, ist der Weg zur Installation einfach. Man lädt das entsprechende Paket herunter, installiert es, und es funktioniert. Die vorkompilierte Version kommt immer mit den Plugins mitgeliefert, die von der Slicer3 Entwicklungsteam bis zu diesem Zeitpunkt integriert wurden.

Die selbstkompilierte Version

Wenn man in Slicer3 entwickeln will, ist dies die Version, die man benötigt. Um den Sourcecode von Slicer3 zu bekommen, muss man ein Shell Command Window öffnen. (Auf Linux & Mac Systemen vorinstalliert, auf Windows Systemen muss diese Software erst heruntergeladen werden, zB: cygwin [2]). und die Befehle

```
svn co
http://svn.slicer.org/Slicer3/branches/Slicer-X-X Slicer3Code
./Slicer3/Scripts/getbuildtest.tcl
```

[2] Cygwin - http://www.cygwin.com/

eingegeben werden. X-X steht hierbei für die Realease Version von Slicer3. 'Slicer3Code' bezeichnet das Directory auf der Festplatte, in das alles heruntergeladen wird. Der untere Befehl führt ein TCL Script aus, das die eigentliche Installation einleitet.

```
getbuildtest.tcl
```

wird mit Standardeinstellungen gestartet, die sich aber auf Wunsch ändern lassen. An diesem Punkt gibt es einige Voraussetzungen die für die verschiedenen Betriebssysteme gegeben sein müssen, damit es hier zu keinen Fehlermeldungen bei der Installation kommt.

Mac Betriebssysteme Für die Installation auf Mac Systemen benötigt man die letzte Version von Apples Entwicklunsgumgebung Xcode [3]. (Nachdem man sich bei Apple registriert hat, kann man Xcode herunterladen. Zustätzlich wird die letzte Version der Versionsverwaltungssoftware Subversion (http://subversion.apache.org/) vorrausgesetzt. *Slicer3 wird auf dem Mac standardmässig in 32 Bit kompiliert.*

Linux Btriebssysteme Für Linux müssen folgende Pakete installiert sein:

- gcc-c++
- gcc
- libX11
- libX11-devel (libX11-dev auf Ubuntu 7.04)
- libXt-devel (libXt-dev auf Ubuntu)
- opengl/mesa (libgl1-mesa-dev auf Ubuntu)
- glu (libglu1-mesa-dev auf Ubuntu)

Slicer3 wird unter Linux Systemen standardmässig mit der Bitrate des Prozessors kompiliert.

[3] XCode - http://developer.apple.com/technologies/tools/xcode.html

Windows Betriesbssysteme Windows Systeme benötigen die meiste Vorarbeit um Slicer3 installieren zu können.

- Developer Studio 9.0 2008 Visual C++ oder darunter. Developer Studio 9.0 2008 Visual C++ ist gratis, und kann von Microsofts Homepage heruntergeladen werden, besitzt aber keinen Debug Modus, und standardmäßig nur einen 32 Bit Compiler.
- Cygwin Hierbei müssen zusätzlich noch die Pakete:
  - curl
  - cvs
  - subversion
  - tcltk
  - unzip
  - zip

  installiert werden.
- NSIS [4]

Slicer3 wird auf Windows Systemen standardmässig in 32 Bit kompiliert.

Solaris Betriebssysteme

- Nur für Open Solaris oder Open Indiana .
- Apache Standard C++ Bibliothek, mit den Paketen SUNWlibstdcxx4 und Bibliothek /c++/stdcxx.

Slicer3 wird unter Solaris Systemen standardmässig mit der Bitrate des Prozessors kompiliert.

[4] NSIS - http://nsis.sourceforge.net/Download

3.3.3 Slicer - Übersicht

Die Basis von Slicer3 kann als Frameworksoftware verstanden werden,d.h Slicer ist ein mehrere Frameworks umspannendes und verbindendes Programm. Diese Frameworks sind es, die Slicer einen Großteil seiner Funktionalität geben, und mit denen der Programmierer arbeiten muss, um seine eigenen Projekte zu entwickeln. Funktionalitäten werden in Slicer3 durch Module implementiert, die Arbeitsprozesse einleiten und steuern. Module werden aus Menüs über die grafische Benutzeroberfläche von Slicer3 gestartet. Slicer3 Module arbeiten mit der MRML (Medical Reality Markup Language) um Arbeitszustände zu definieren.

MRML

MRML (Medical Reality Markup Language[5]) ist ein Markuplanguage Dateiformat (ähnlich wie XML) zum Speichern von medizinischen Sachverhalten. In Slicer3 wird MRML verwendet um Module zu verwalten und unter anderem, um die Prozessabfolge von Userbefehlen zu beschreiben. Auf diese Weise kann der Benutzer in verschiedenen Modulen Einstellungen an Algorithmen, Ansichten, und Daten vornehmen, sowie semantische Information hinzufügen, diesen Bearbeitungsstandpunkt abspeichern, bzw. Arbeitsschritte rückgängig machen. Dadurch lassen sich z.B: dieselben Einstellungen an zwei unterschiedlichen Instanzen von Slicer3 zu einem Datensatz vornehmen. Das MRML Protokoll erlaubt es Slicer ausserdem, Module aufzulisten und intern für einander sichtbar zu machen. Somit können Module auch miteinander agieren, und eine Verarbeitungspipeline zu bilden.

Die grafische Benutzeroberfläche (GUI)

Nach dem Start von Slicer3 öffnet sich das Hauptfenster. (Abb. 3.10) Grob lässt es sich in 6 Funktionseinheiten zerlegen.

1. Die Modul Auswahl: Alle Slicer3 Module werden hier aufgeführt. Ein Modul lässt sich einfach mit einem entsprechenden Mausklick laden.
2. Das Modul Fenster: Dieses Fenster zeigt das Hauptfenster des gewählten Moduls an. Jedes Modul besteht aus einem oder mehreren in sich verschachtelter Fenster, die zu und aufklappbar sind.

[5] MRML - http://www.mrml.net/

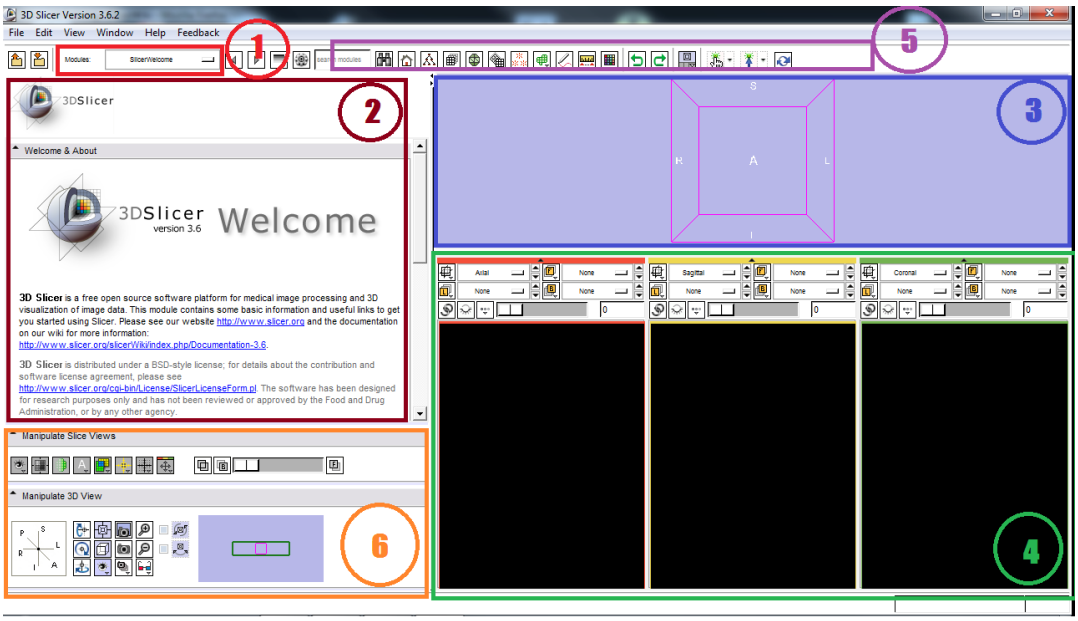

Abbildung 3.10: Das Slicer Hauptfenster.

3. Das 3D Render Window: Zeigt das 3D Modell des geladenen Datensatzes an.

4. Die Planaransicht: Zeigt axial, sagittal und coronale Ansichten des Datensatzes an. Hier kann man die verschiedenen Schichten des Datensatzes durchblättern, sowie sich die Lage der Schicht im 3D Render Window anzeigen lassen. Die Darstellung in der Axialplanaransicht ist eine Rekonstruktion der Originaldaten in einem 3D Würfel. Bei schiefliegenden Daten stimmt deshalb die Ansicht einer Schicht nicht mit dem Aussehen der entsprechenden Schicht im Datensatz überein.

5. Modul Schnell Lade Leiste: Ein Leiste für gebräuchliche Module.

6. 3D Edit Box: Hilft spezielle Ansichten von dem 3D Render Window zu erstellen (Kamera ändern, Snapshots, usw.).

- File: Zum Laden / Speichern von Daten. Slicer3 kann eine Vielzahl von Datenformaten lesen. DICOM Bilder können über das Menü $File \rightarrow Add\ Volume$" gelesen werden. Alle anderen gängigen Bildformate (u.a tiff, jpeg, nrrd, vtk) werden über den Menübefehl $File \rightarrow Add\ Data$" geladen. Alle verfügbaren Datenformate lassen sich außerdem in jedem anderen Format speichern. Mit Slicer3 lassen sich zudem Markup Language Files (xml, mrml) laden, um Daten zu beschreiben.

- Edit: Undo / Redo Funktionen, Längenmessungen.

- View: Optionen rund um das Laden und Installieren von Modulen, sowie verschieden Systemoptionen.
- Window: Verändert die Ansichten des Hauptfensters.
- Help
- Feedback

3.3.4 Module

Alle relevanten Arbeitsprozesse werden in Slicer3 über Module gesteuert. Die Basisinstallation von Slicer3 hat bereits eine große Anzahl von Modulen (Vers 3.6 über 50), und es kommen ständig neue dazu. Module sind prinzipiell frei einbaubare Elemente, die meist von verschiedenen Personen oder Gruppen programmiert worden sind. Da Slicer3 auf dem Prinzip der freien Softwareverwendung publiziert wurde kann jeder, der sich ein bißchen mit C, C++ auskennt mehr oder weniger aufwändige Module schreiben. Dies bedeutet, dass die Verwendung von Modulen stets auf eigene Gefahr geschieht. Das Slicer3 Maintainance Team nimmt keine semantischen Überprüfung auf Richtigkeit und Funktionalität der Module vor. Zu beachten ist auch dass der Sourcecode von Modulen nicht unter einem freien Verwendungskontext stehen muss. Module die über die Menüleiste unter dem Punkt View heruntergeladen werden können nennt man Extensions. Extensions lassen sich auf jeder Version (selbstkompiliert,precompiliert) von Slicer3 starten. Selbstgeschriebene und kompilierte Module werden über den Punkt der Menüleiste "$View \rightarrow Application\ Settings \rightarrow Module\ Settings$" geladen. Da die Dokumentation einzelner Module am besten über deren Hilfstext zu entnehmen ist, und es eine ständig wachsende Anzahl von Modulen gibt, folgt hier nur eine schnelle Grobeinteilung der bis dato verfügbaren Module.

Dies sind die wichtigsten Grundelemente der Slicer3 Software. Da die Bemühungen alle Funktionalitäten und Fenster zu erklären den Rahmen dieser Arbeit leicht sprengen würden, sei der interessierte Leser hiermit auf die Online Dokumentation von Slicer 3.6 verwiesen.

Basismodule (Color, Data, Fiducials, Measurements, Models, ROI, Slices, Transform, Volume Rendering, Volumes) sind alles vom Slicer Entwicklungsteam programmierte Module. Sie bieten Möglichkeiten geometrische Messungen vorzunehmen, 3D Modelle zu erstellen,

Transformationen anzuwenden, oder sich den Verlauf der Arbeitsschritte mit dem MRML Data Model anzuschauen. Mittels des ROI Moduls können rechteckige Block Regionen im 3D Modell angezeigt werden, die als Input für andere Module dienen können.

Spezialmodule Spezialmodule sind verschieden Gruppen zugeordnet.

- Wizards:
  Wizards sind angeleitete Module. Das Programm lässt Benutzer eine Folge von Eingaben machen und erledigt den Rest von selbst.
- Informatics:
  Informatics Module durchsuchen häufig Datenbanken nach den Eingaben des Benutzers.
- Registration:
  Slicer3 bietet bei der Registration verschiedener Bildmodalitäten die Möglichkeit rigid, non rigid, affine und manuelle Registration durchzuführen. Zusätzlich dazu gibt es einige Module für Spezialfälle, etwa Hirn MRI Bilddaten, Registration aufgrund von Oberflächenübereinstimmung, oder Registration aufgrund von Erkennungsmarkern.
- Segmentation:
  Zur Segmentierung gibt es in Slicer Module, die folgende Algorithmen benutzen: Region Growing Algorithmen (treshhold basiert, und Minimum Entropy) .
- Quantification:
  Beinhaltet ein Modul zur Quantifizierung einer Krebsbehandlung.
- Diffusion:
  Slicer3 bietet viele Module zur Behandlung von Diffusion MR Bildern. Es gibt 3 Unterkategorien:
  1. 'Denoising Module': Helfen die Bildqualität von DWI Bildern zu verbessern.
  2. 'Diffusion Tensor Utilities': Alle Arten von Berechnungen auf DWI Tensoren.
  3. 'Tractography': Versuchen Hirnregionen mit den DWI Bildern in Übereinstimmung zu bringen.
- IGT:
  Mehrere Module zu Image Guided Therapy (IGT).

- Filtering:
  Filtermodule aus den Kategorien Treshholdfilter, Glättungsfilter, Morphologiefilter, oder Additions,und Subtraktionsfilter.
- Surface Modelling: Module zur Erstellung von Modellen aus Datensätzen und deren grafischer Darstellung.
- Converters: Konvertieren Datensätze eines Formats in ein anderes.
- Endoscopy: Ein Modul zur virtuellen Endoskopie.
- Developers Tools: Diverse Tool Module.

Die obige Auflistung soll nur einen groben Überblick über die verfügbaren Module geben. Eine genauere Beschreibung findet sich in der online Dokumentation von Slicer 3-6.

Neue Module

In Slicer können neue Module heruntergeladen (Extensions) , oder eigens kompilierte integriert werden. Letztere können Loadable Modules, oder Command Line Processing (CLP) Module sein. Loadable Modules werden als Dynamic Link Libraries abgespeichert (dll) und können nur auf solchen Systemen verwendet werden, auf denen Slicer mit dem selben Kompiler kompiliert wurde, wie das Modul selbst. Ein freier Import solcher Module ist nicht ohne Einschränkungen auf andere Slicer Versionen, die mit anderen Compilern kompiliert wurden, möglich. Die Entwickler verwenden für die vorkompilierte Version von Slicer3 einen eigenen C++ Compiler, mit dem auch alle bis dato verfügbaren Module kompiliert werden, und somit für alle vorkompilierten Versionen von Slicer ohne Einschränkungen verfügbar sind. CLP Module hingegen werden als ausführbare Dateien kompiliert, und sind ohne Einschränkungen überall verwendbar.

1. Extensions:
   Extensions können in Slicer über den Menüpunkt "$View \rightarrow Extension\,Manager$" heruntergeladen und installiert werden werden. Jedes Modul besitzt einen Informationstext, der angibt wie und wann das Modul eingesetzt werden kann. Extensions sind Module, die bereits vom Slicer Entwicklungsteam zur Importation freigegeben wurden, und alle dem Slicer3 MRML Execution Model für Module entsprechen.

2. Loadable Modules:
   Loadable module sind eigens kompilierte Module und werden über den Punkt "$View \rightarrow Application\ Settings \rightarrow Module$" geladen.

3. Scripted Modules:
   Module können auch in einer Scriptsprache wie TCL oder Phyton programmiert werden. Gescriptete Module werden zur Laufzeit von Slicer3 mit einem eigenen Interpreter ausgeführt.

4. Command Line Processing Modules (CLP):
   CLP sind eigens kompilierte Module, die aus einer ausführbaren Datei bestehen, die mit einem XML File beschrieben werden, um den Datenfluss mit Slicer zu regeln, und das GUI des Moduls zu kreieren. CLP Module können auch als selbstständige Programme laufen, und werden zur Slicer3 Laufzeit ausgeführt. Sie haben eine eingeschränkte Interaktion mit dem Benutzer und den Nachteil, dass sie nicht auf Slicer3 Systemvariablen zugreifen können, oder im MRML Profil aufgelistet werden können. Andere Module können somit nicht auf CLP Module zugreifen.

3.4 Programmieren in Slicer

3.4.1 Systemarchitektur

Slicer3 ist ein Programmframework aus über einer Million Zeilen Code, größtenteils C++, aber auch mit Schnittstellen zu TCL oder Phyton. Slicer verwendet selbst Frameworks, wie das Insight Development Kit (ITK), das Visualisation Kit (VTK), sowie KWWidgets. Je nachdem welche Art von Modul man programmieren will, desto tieferes Wissen über die einzelnen Frameworks sollte man mitbringen. Der Buildprozess wird in Slicer3 von Cmake [6] übernommen. Cmake ist ein Cross-Platform Programm zum Zusammenschnüren von rohen Sourcecode Projekten, mit dem makefiles generiert werden, die von einem lokalen Compiler gelesen und compiliert werden. Cmake wird ausserdem für Loadable und CLP Module benötigt, da diese selbst kompiliert werden müssen. Der nächste Abschnitt setzt sich näher damit auseinander wie CLP und Loadable Module in Slicer geschrieben werden.

[6] CMake - http://www.cmake.org/

Slicer Execution Model Jedes Slicer Modul, egal ob CLP basiert, loadable oder scriptable muss sich an die Richtlinien des Slicer Execution Models halten. Für CLP Module und gescriptete Module übernimmt die Erstellung aller notwendigen Elemente das XML File. Bei loadable Moulen bleibt es dem Entwickler überlassen diese selbst zu implementieren. Das Slicer Exekutions Model definiert 3 Einheiten (Logic, MRML Knoten und GUI)), die den Datenfluss, die logische Abarbeitung und ihre Relationen zueinander festlegen. Das Zusammenspiel von MRML Knoten, Logic und GUI wird in Abb. 3.11 genauer gezeigt. Die GUI jedes Modells schickt Eingaben des Benutzers an den MRML Knoten, der von Slicer3 verwaltet wird. Üblicherweise besitzt jedes Modul eine andere Art von MRML Knoten, ist also einer anderen C++ Klasse zugeordnet. Der MRML Knoten kann als Datenreservoir aller Einstellungsparameter verstanden werden, und speichert immer den letzten Parameterstand der GUI. MRML Knoten sind überwachbar, dh. verschiedene Funktionen können auf Datenänderungen innerhalb eines MRML Knotens reagieren und getriggert werden. Zwischen GUI und MRML Knoten steht Logic. Logic steht mit allen Arbeitsaufgaben in Verbindung und ist funktional von der GUI und dem MRML Knoten losgelöst. Logic kommuniziert mit dem MRML Knoten, berechnet Aufgaben mit den gespeicherten Parametern, und setzt diese Parameter neu am Ende der Berechung. Die GUI reagiert auf Änderungen im MRML Knoten, entnimmt alle notwendigen Parameter und setzt diese in der GUI auf ihren neuen Wert. Abb. 3.11 zeigt, daß diese Verarbeitunsgpipeline multidirektional ist; vom Prinzip kann jede Einheit mit einer anderen kommunizieren. Somit ist der Programmierer nicht komplett an die Ausführung diese Models gebunden. Funktionen die z.B: mit Logic zu tun haben können eigentlich auch direkt im MRML Knoten definiert werden. Obiges Modell ist eher als eine Designvorlage der Entwickler zu verstehen, um Slicer3 Module transparenter und gekapselter zu gestalten.

CLP Module schreiben

Die einfachste Modul Form ist das CLP Modul. Dies sind Module die von Slicer3 über die Commandline Eingaben bekommen, diese abarbeiten und das Ergebnis dann an Slicer3 zurückschicken. Module dieser Form müssen einer XML spezifischen Syntax folgen um zu funktionieren.

Die CLP XML Syntax

XML Files, die CLP Programme beschreiben sollen, müssen alle folgende Minimale Struktur aufweisen:

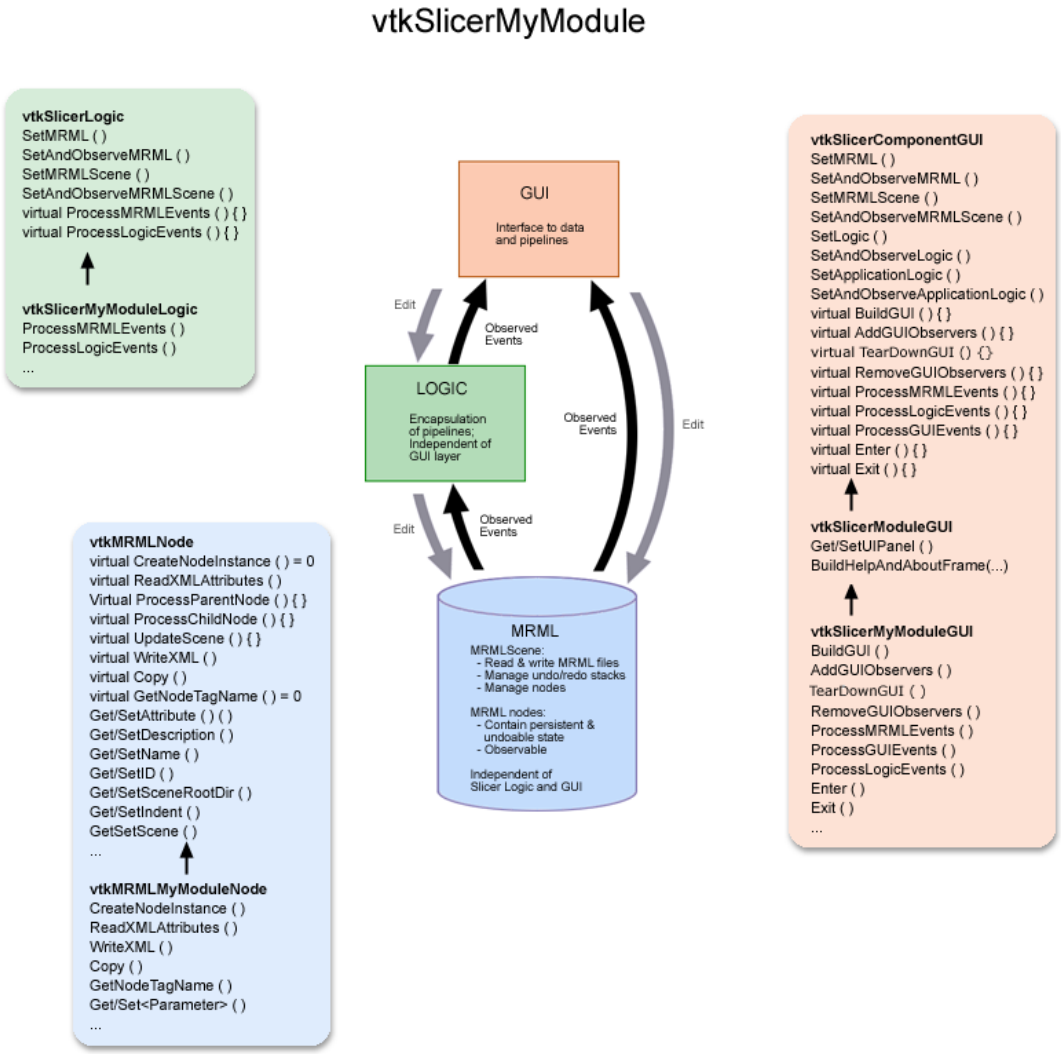

Abbildung 3.11: Das Slicer Execution Model. [12]

```
<?xml version="1.0" encoding="utf-8"?>
<executable>
<title> A title </title>
<description> A description </description>
  <parameters>
   At least one parameter
  </parameters>
</executable>
```

<title> Gibt den Namen des Moduls an. <description> Gibt eine Beschreibung des Moduls an, der unter Help & Aknowledgements zu finden ist.
<parameters> Im XML können alle arithmetischen Basistypen definiert werden, sowie einige Datentypen wie String, Vector oder Bilder. Folgendes Beispiel legt einen Integer Vector an:

```
<integer-vector>
<name>Vectorname</name>
  <longflag>--Vectorname</longflag>
<description>Description of the Vector</description>
  <label>Vector label shown</label>
  <default>1,1,6</default>
</integer-vector>
```

Ein Bild hingegen wird so definiert:

```
<image>
<name>inputVolume</name>
   <label>Input Volume</label>
   <channel>input</channel>
   <index>0</index>
   <default>None</default>
   <description>Input volume</description>
</image>
```

<channel> Definiert ob ein Bild ein Eingabe oder ein Ausgabe Bild ist.
<index> Startet bei 0, Gibt die Stelle des Elements in der Command Line an. Werden vom

Benutzer zuwenig Eingaben gemacht um bis zum Index des Objektes zu gelangen, so wird das Modul nicht gestartet.

Slicer3 liest zur Laufzeit das XML File jedes verfügbaren CLP Modules, generiert daraus einen MRML Knoten, sowie das Modul GUI, und legt alle notwendigen Objekte an, die dann dem Modul übergeben werden. Abb. 3.12 zeigt ein Beispiel GUI wie man es durch ein entsprechendes XML File kreieren könnte. Da es noch viele Befehle mehr gibt was man alles in dem XML file definieren kann, sei der Leser auf die Slicer Dokumentation verwiesen. Jedes CLP Modul besitzt ausserdem defaultmässig einen 'Apply' Button, mit dem die entsprechende Aufrufmethode (main) der ausführbaren Datei gestartet wird. Diese Art von Modulen ist sicherlich die schnellste Möglichkeit etwas in Slicer3 umzusetzen, da man hier eigentlich nur die gewünschten Algorithmen implementieren muss. Die Programmierung der GUI wird durch das XML File auf einen Minimalaufwand reduziert. CLP Module können ausserdem unabhängig von Slicer gestartet und getestet werden. Dies schränkt allerdings auch die Verwendung von CLP Programmen für Slicer ein. Command Line Module haben keine Übersicht über die MRML Szene und können auf keine Daten von Slicer zugreifen. Die einzigen Daten, die sie beziehen können stammen durch die Command Line Eingabe. Module dieser Sorte bieten wenig Interaktivität mit dem Benutzer, und sind am besten für Aufgaben geeignet, die sich auf den ganzen Datensatz beziehen.

Loadable Module schreiben

Loadable Module werden in Slicer als Dynamic Link Libraries (dll) Files zum Systemstart geladen, und werden in die MRML Szene von Slicer3 integriert. Dies ermöglicht Modulen auf Systemvariablen von Slicer zuzugreifen, sowie auf andere Module, und die Möglichkeit diese Aufzurufen, bzw zu interagieren. Loadable Module sollten vom Entwickler nach den Richtlinien des Slicer3 Exekutionsmodells geschrieben werden. Dazu muss jedes Slicer Loadable Module 3 Basisklassen definieren:

1. vtkSlicerMyModuleGUI: Jede GUI Klasse muss von der Slicer3 Klasse vtkSlicerModuleGUI abgeleitet sein. Wenn Slicer ein Modul startet wird zunächst die Funktion $Enter()$ aufgerufen, die im nächsten Schritt die GUI zusammengebaut, ($CreateGUI()$) sowie die Funktion $UpdateGUI()$ ausführt, die die GUI Elemente (Widgets) mit den Parameterstartwerten im MRML Knoten befüllt. Wird das Modul geschlossen, so wird die Funktion $Exit()$ aufgerufen, welche alle Objekte wieder freigibt. $Enter()$ und $Exit()$

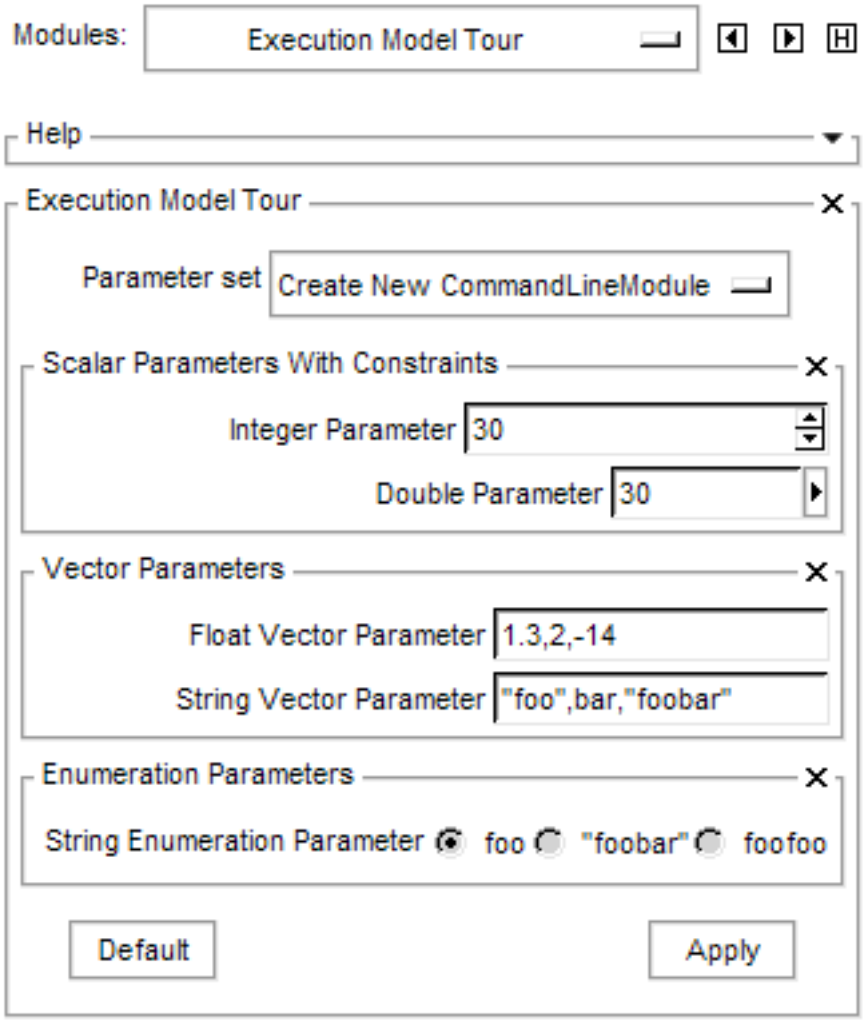

Abbildung 3.12: Slicer Beispiel CLP Modul GUI [8]

sind statische Methoden und dürfen nicht verändert werden. Folgende Methoden müssen vom Entwickler ausformuliert werden:

- CreateGUI: Das komplette Aussehen des GUI wird hier festgelegt. Bauelemente sind vtkKWWidgets.
- DestroyGUI: Wird beim Schließen des Moduls aufgerufen. Ruft den Deallocator von vtkSlicerMyModuleGUI auf. Bei der Methode $DestroyGUI()$ sollte der Entwickler beachten, dass jedes mit dem Befehl " $\rightarrow New()$" angelegte Objekt wieder mit dem Befehl " $\rightarrow Delete()$" gelöscht werden muss, da sonst beim Schließen von Slicer3 Fehler auftreten. Beim Löschen von Widgets ist weiters die Reihenfolge beim Löschen zu beachten; z.B: sollte ein Textlabel in einem Reiter vor dem Löschen des Reiters gelöscht werden.
- ProcessGUIEvents: Reagiert auf Events von Widgets oder GUI Klassen. Leitet Daten von der GUI an den MRML Knoten weiter.
- ProcessMRMLEvents: Reagiert auf Veränderungen im MRML Knoten. Leitet Daten vom MRML Knoten an Widgets weiter.
- AddGUIObservers: Legt Beobachter zu Widgets fest.
- AddMRMLObservers: Legt Beobachter zu MRML Knoten fest.
- RemoveGUIObservers
- RemoveMRMLObservers
- UpdateGUI: Kopiert den Status der MRML Knoten in das GUI.
- UpdateMRML Kopiert den GUI Status in den MRML Knoten.
- vtkSlicerMyModuleLogic: Jede Logic Klasse muss von der Klasse vtkSlicerModuleLogic abgeleitet werden. LogicKlassen sollten aus Design und Kapselungszwecken alle Funktionen enthalten, mit denen der User in der GUI einen Aufgabenprozess starten kann. Die Logic Klasse ist von der MRML Szene, sowie der GUI unabhängig, was es dem Prorammierer erleichtert die Funktionsweise der Algorithmen zu überprüfen. In der Klasse Logic können neue MRML Knoten in die Slicer3 Szene sowie MRMLObserver eingebaut werden, die auf Ereignisse aus anderen Knoten reagieren können. Auf diese Weise ist z.B: eine Kommunikation zwischen zwei Modulen möglich.

module/algoritjmusdiagrammloadable.jpg

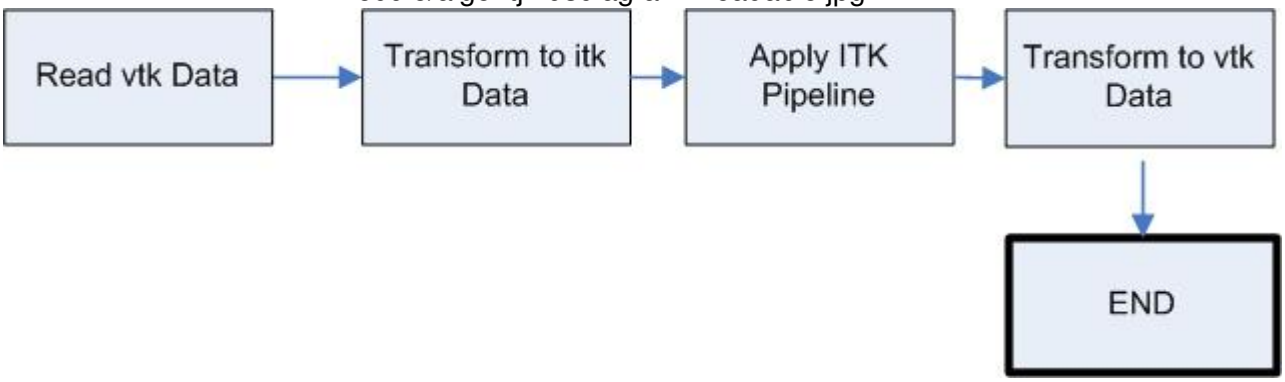

Abbildung 3.13: Ablauf des Slicer3 Loadable Moduls mit der ITK Pipeline aus Absch. 3.2.1

- vtkMRMLMyModuleNode:

  Jeder MRML Knoten muss von der Klasse vtkMRMLNode abgeleitet werden. In dieser Klasse werden alle Parameter festgelegt die auch in der GUI einstellbar sind. Weiters können Objekte definiert werden, die z.B: in der Klasse Logic benötigt werden. MRML Knoten stehen zwischen der Logic Klasse und der GUI Klasse, und dienen quasi als Datenaustauschrservoir für beide. Weiters bietet die Klasse vtkMRMLNode die Möglichkeit XML Files zu lesen (wird zb im Falle eines CLP basierten Modules gemacht um den MRML Knoten zu definieren) und XML Files zu schreiben, und somit alle Atribute des MRML Knotens festzuhalten.

3.4.2 Umsetzung des MFP Moduls

Das MFP Plugin wurde in Slicer3 sowohl als CLP Modul, als auch als loadable Modul auf Basis der ITK Version des Algorithmus aus Absch.3.2.1 implementiert. Im Verlauf der Arbeit war zunächst nur das CLP Modul eingeplant, es stellte sich jedoch schnell heraus dass ein CLP Modul nicht den Anforderungen genügen würde, und so wurde auch ein entsprechendes Loadable Modul entwickelt. Entwickelt wurde auf einem Intel i7 2,8GH Quadcore mit 8GB Ram, Windows7 PE. Da mit der Freeware Version von Visual Studio 2008 entwickelt wurde, mussten alle Dateien im 32 Bit Modus kompiliert werden. Abb. 3.13 gibt eine Übersicht über den Ablauf des Algorithmus des loadable Moduls , Abb.3.14 über den des CLP Moduls. Wie man sehen kann sind beide bis auf ein paar Konvertierungsprozesse zwischen dem in Slicer3 gebräuchlichen Datenformat (vtkImageData) und dem in der ITK Pipeline verwendeten itkImage, gleich.

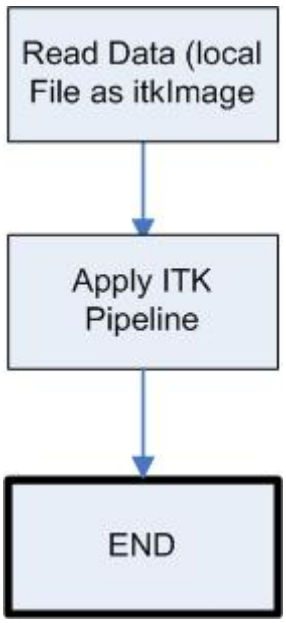

Abbildung 3.14: Ablauf des Slicer3 CLP Moduls mit der ITK Pipeline aus Absch. 3.2.1

CLP Modul

Das MFP CLP Modul ist eine einfache Anwendung (Abb. 3.15), bei der der Benutzer die Werte für die Gausschen Standardabweichungen, sowie die Multiplikationsfaktoren eingeben kann. Wenn alle Felder ausgeführt sind, klickt er auf 'Apply', und der gesamte Datensatz wird mit den Einstellungen berechnet. Der große Nachteil des Moduls ist es, dass es keine Previewansichten der Filterform, sowie der Bilder besitzt, und den Benutzer somit relativ im Unklaren lässt was er gerade tut.

Loadable Modul

Das Konzept des Plugins ist es, dass der Benutzer über mehrere Regler die Standardabweichungen der Gausskurven bestimmen kann. Damit er verschiedene Einstellungen testen kann, und nicht immer auf den ganzen Datensatz anwenden muss, was mitunter sehr zeitintensiv werden würde, sucht er sich eine Slice im Datensatz aus, (die erste bei einzelnen Bildern , wie z.B Röntgen), die dann als Vorschaubild für alle vorgenommen Veränderungen fungiert. Zu jedem verwendeten Regler wird eine Voransicht des entsprechenden Differenzbildes generiert. Hat der Benutzer die Wahl für seine Bänder getroffen, muss er noch die Multiplikationsfaktoren der einzelnen Bänder wählen. Das Summationsbild wird daraufhin er-

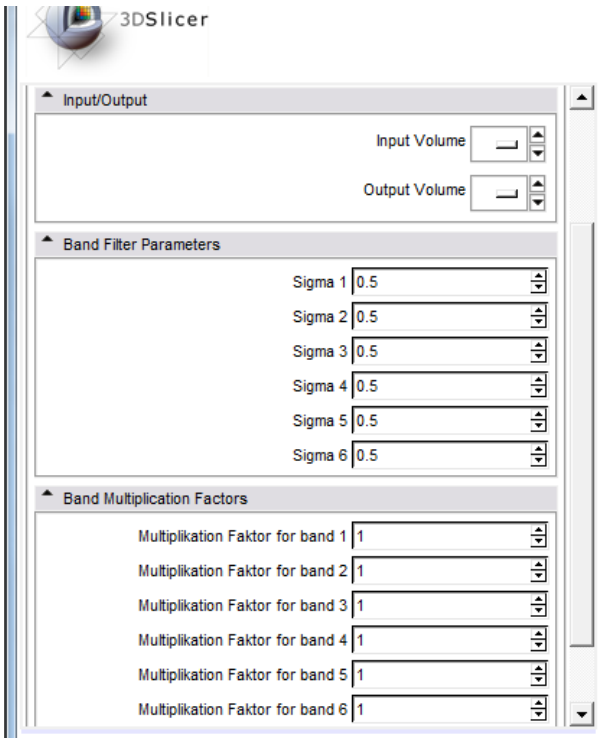

Abbildung 3.15: MFP CLP Modul für Slicer3

stellt, und gibt Aufschluss darüber wie der Datensatz nach Bearbeitung aussehen würde. Ist die Parameterwahl zufriedenstellend, können die Einstellungen für den ganzen Datensatz übernommen werden. Abb. 3.16 zeigt in einem Use Case Diagramm an, was der Benutzer alles machen kann. <include> steht dabei für jene Anwedungsfälle die vor dem gerade betrachteten erledigt werden müssen. Die Wahl für die Anzahl der Bänder ist nicht frei wählbar. Da Slicer3 entweder xml benötigt um die Plugin Gui zusammenzubauen, oder eine GUI Klasse, wird shon zum Start von Slicer3 das genaue Aussehen jedes Moduls statisch festgelegt. Aus diesem Grund wurde die Anzahl der Bänder willkürlich auf maximal 6 festgesetzt.

Programmierung

Die GUI Klasse:

Wie bereits erwähnt ist der linke Teil des Slicer3 Hauptbildschirms für das Menüfenster von Plugins reserviert. Da der Platz hier sehr knapp bemessen ist musste ein Kompromiss aus Übersicht und Darstellung geschlossen werden. Hätte man alle Beispielbilder nebeneinander gestellt, wäre das zum Nachteil ihrer Größe ausgefallen. Das Plugin wurde nun in die Länge gezogen. Beim Start des Moduls öffnet sich der Hauptbildschirm (Abb. 3.17). Punkte 1-5

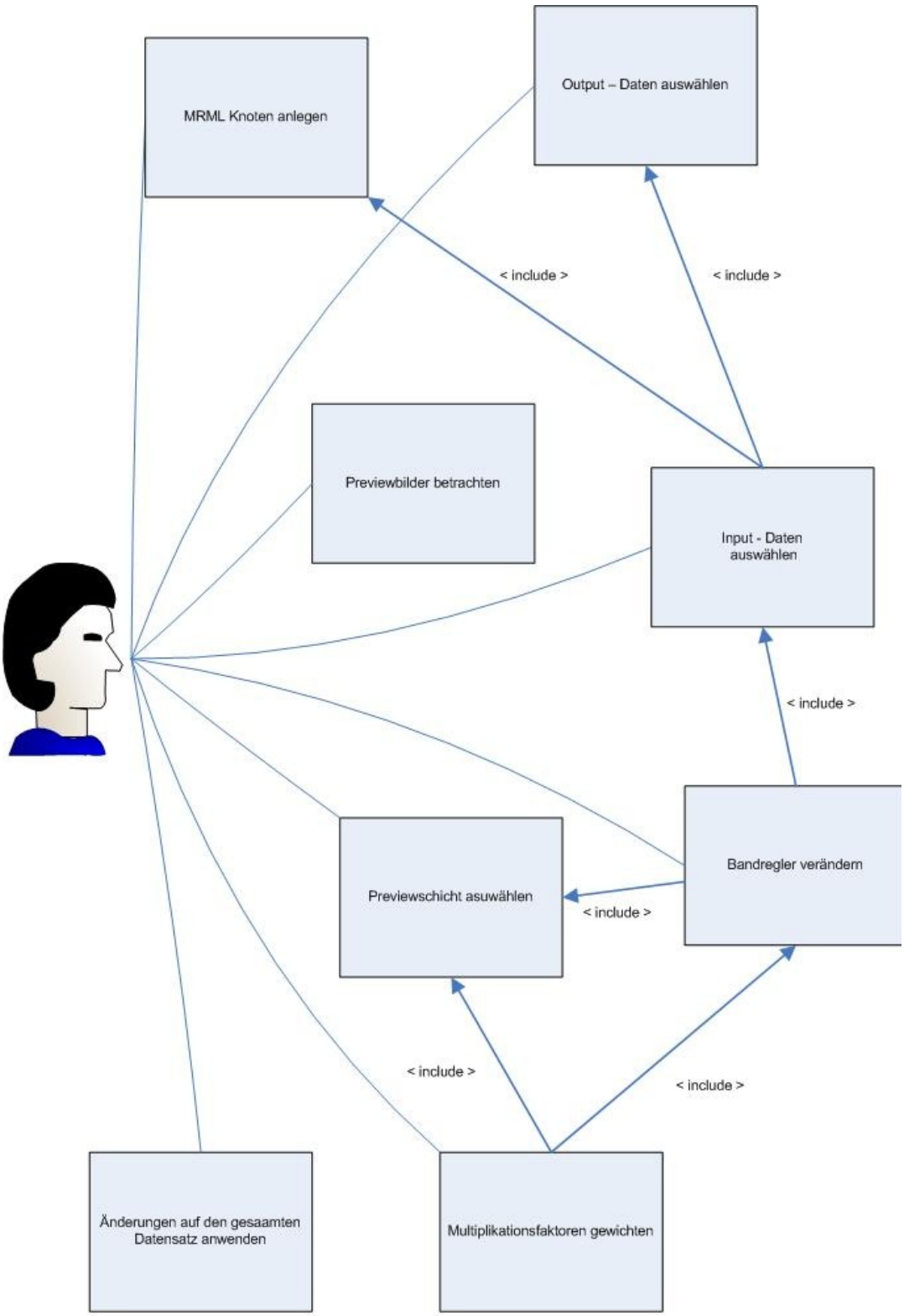

Abbildung 3.16: Use Case Diagramm des Slicer3 Loadable Moduls

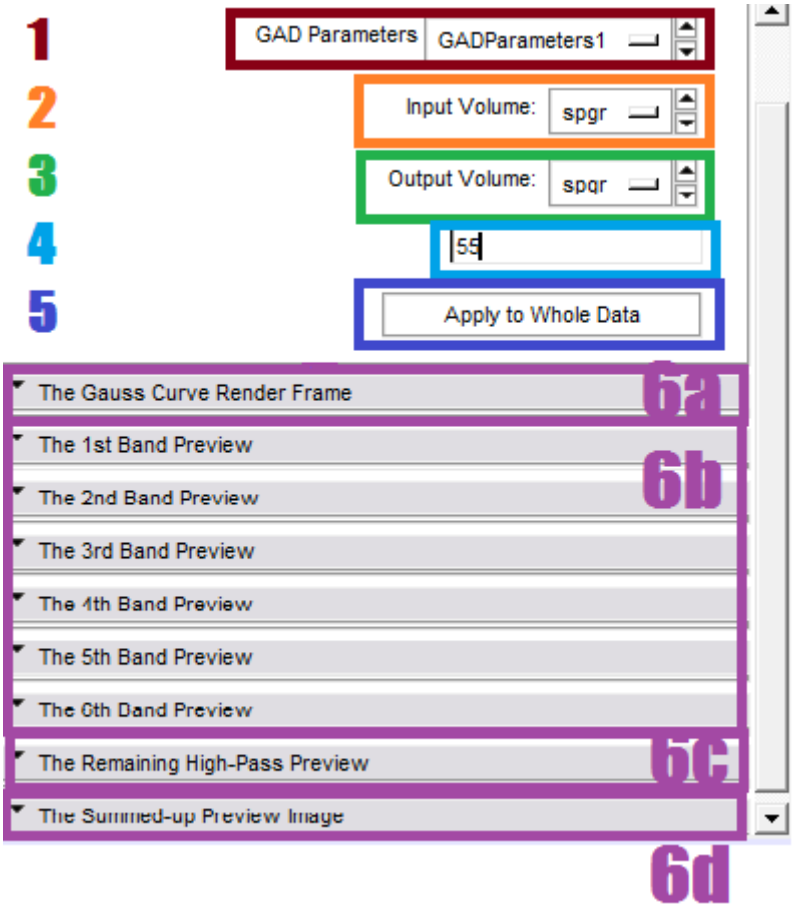

Abbildung 3.17: Slicer3 MFP Loadable Modul GUI

kann man als Setup Parameter bezeichnen, Punkte 6 a-d sind für die visuelle Darstellung und Parameterwahl des Algorithmus.

1. GAD Parameters:
   Das ist die Dropdown Auswahl für den derzeit ausgewählten MRML Knoten. Wenn das Modul geladen wird, bevor ein Datensatz geladen wurde, ist die Dropdown - Leiste leer. Das Laden eines Datensatzes triggert die Erstellung eines neuen MRML Knoten. Wurden bereits durch andere Module MRML Knoten erstellt, und wurde eine ensprechende Formatierung durchgeführt um den speziellen Knoten zu lesen, können über diese Schnittstelle MRML Knoten miteinander kommunizieren.

2. Input:
   Hier wählt der Benutzer den zu verwendenen Datensatz aus. Wurde im Slicer Hauptbildschirem ein Datensatz geladen, muss man ihn hier noch zusätzlich auswählen. Das

Auswählen eines Datensatzes wiederum triggert die Erstellung eines neuen MRML Knoten aus Punkt 1, falls noch kein aktiver exisitiert. In Slicer3 werden Bilddaten intern immer im virtuellen VTK Datenformat vtkImageData gelesen. Um die Bilddaten für die ITK Pipeline vozubereiten mustse eine Konvertierung von vtkImageData zu itkImage vorgenommen werden. Die Konvertierung funktioniert jedoch nur für folgende Datentypen, wie sie in vtkSetGet.h definiert werden:

- VTK_CHAR
- VTK_UNSIGNED_CHAR
- VTK_SHORT
- VTK_UNSIGNED_SHORT
- VTK_INT
- VTK_UNSIGNED_INT
- VTK_LONG
- VTK_UNSIGNED_LONG
- VTK_FLOAT
- VTK_DOUBLE

Sollte der geladene Datensatz nicht einem der obigen Formate entsprechen, wird eine Fehlermeldung ausgegeben.

3. Output:
   Die Auswahl für den Output. Nach Anwendung des Filters befindet er sich im FLOAT Format.

4. Anzeige für die PreviewSlice:
   Hier gibt der Benutzer die Nummer der Slice an, die er angezeigt bekommen will. Wenn die eingegebene Zahl zu groß ist, wird die letzte Schicht angezeigt.

5. Wendet die Visuellen Parameter auf das gesamte Datenset an.

6. **(a), Gausskurven Render Frame:**
   Hier werden die Standardabweichungen für die Gaussschen Filtermasken im Fourierraum ausgewählt. Um einen besseren visuellen Eindruck zu erhalten wird dem Benutzer das 2D Aussehen der gerade aktiven Wellenfront mit Markierungen der Maximas

auf der Xachse, sowie die Summe über alle Gaussbänder angezeigt(Abb.3.19). Jedes Band besitzt dabei eine farbliche Codierung, der Hochpass ist grau, und die Summe aller Teilbänder schwarz. Der genaue Abstand zum Mittelpunkt wird ihm darunter in Linienpaaren / mm angegeben. Leider wurde bei der Implementation der Grafik eine vtkKlasse verwendet, die selbstständig die Achsen verlängert, weswegen hier der optische Eindruck der 'abgeschnittenen' Funktionen ergibt. Das Ende des Datensatzes befindet sich bei ungefähr 0.7 Linienpaaren/mm, auch wenn dies nicht genau angezeigt wird. Die X Achse der obigen Grafik stellt dabei den Querschnitt durch die Gausskurve entlang der ersten Dimensionsachse (Bildmatrixspalten) dar. Der Ursprung 0 befindet sich in der Bilmitte. Die Ränder, der Grafik stellen die maximal erreichbaren Linienpaare / Millimeter (nach Formel 2.12) dar. Diese Information wird aus dem Dicom Header Eintrag [0028,0030 PixelSpacing] gewonnen. Da die Gaussfilter immer breitere Ausdehnung haben, man aber nur eine diskrete Einstellung mit den Reglern vornehmen kann, wurden die Einheiten auf den Reglern so umgerechnet, dass sie bei höchster Einstellung eine Gausskurve erzeugen, die an den Bildrändern noch 40% der Gesamtintensität aufweist.

(b), Preview Ansichten:

Die Previewansichten werden dem Benutzer in durchnummerierten Fenstern, die sich auf Wunsch öffnen und schließen lassen, angezeigt. Jeder der sechs Regler im Gauss Render Frame verändert dabei sein entsprechendes Previewbild, und zeigt das gaussgefilterte Bild bzw. die Subtraktionsbilder der immer größer werdenen Gausskurven an. Wird der Regler wertegleich mit dem genau darüberliegendem eingestellt, ergibt sich als Subtraktion von zwei gleichen Gausskuren ein schwarzes Bild. Jeder der Previewframes besitzt eine frei wählbare Einstellung für die Gamma Kurve.

(c), Hochpassbild:

Der verbleibende Hochpass ist in der 2D Grafik in Grau angezeigt, und gibt den Rest des Frequenzbereiches an, der dem Filter bei einer neutralen Bandgewichtung von 1 fehlt, um das Originalbild wieder herstellen zu können. Der Hochpass Renderframe kann dabei ähnlich wie die Bandpreviewbilder grauwertgefenstert und in der Helligkeit verändert werden.

(d), Summationsbild:

Im Fenster 6d (Abb. 3.18) wird das Summationsbild aller bestehenden Teilbilder berechnet und mit den Multiplikationsfaktoren gewichtet. Die Standardeinstellungen sehen vor, jedes Bild mit eins zu gewichten. Das Ergebnis dieser Einstellung ist das Ursprungsbild. Duch Verschieben der Regler kann der Benutzer die Faktoren jedes

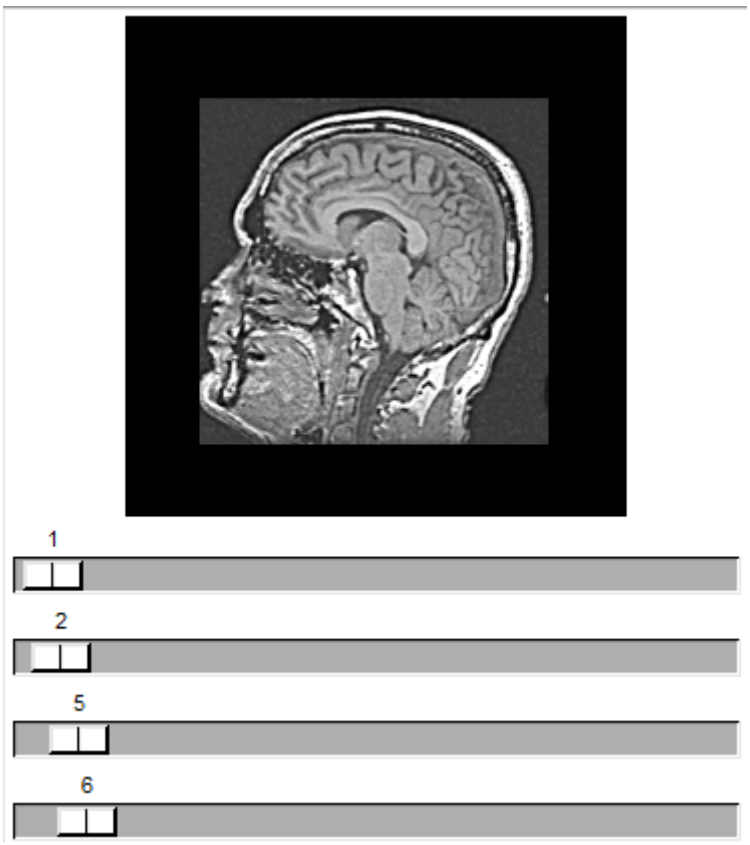

Abbildung 3.18: Summationsbild mit Gewichtung der ersten 4 Bänder.

Bandes verändern, damit das spezielle Teilbild im Endergebnis verstärkt (bis zu maximal 100-fach) oder abgeschwächt wird. Da die Intensitätswerte höherer Bänder immer kleiner werden müssen deren Multiplikationsfaktoren entsprechend gewählt werden, um sichtbare Unterschiede im Summationsbild hervorzuheben. Im Summationsbildframe befindet sich ausserdem eine Check Box mit der der Benutzer den verbleibenden Hochpass ein und ausschalten kann. Dies ist nützlich um den Anteil besser abzuschätzen den der Hochpass noch am Summationsbild ausmacht.

Die MRML Klasse

In der MRML Klasse befinden sich die gespeicherten Werte für

- Die Band Regler.
- Die Multiplikationsfaktoren.
- Den logischen Wet zum Ein / Ausshalten des Hochpasses.

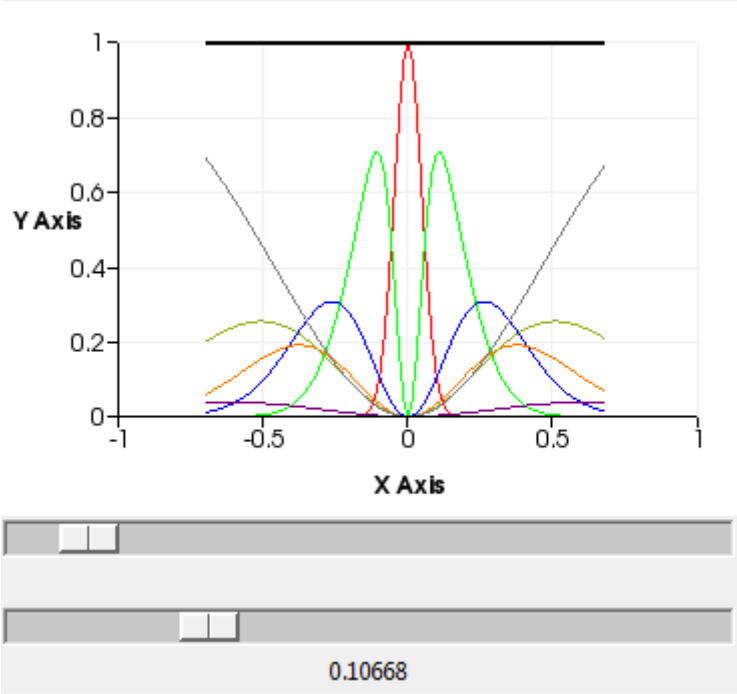

Abbildung 3.19: Gausskurven Render Frame des Slicer3 Loadable Moduls. Leider wird die X Achse von der verwendeten VTK Klasse selbstständig erweitert, obwohl die Daten garnicht soweit reichen, was den Eindruck der ’abgeschnittenen’ Funktionen vermittelt

- Referenzen zu Input und Output Daten.

Die MRML Klassenimplementation dient somit als reiner Platzhalter für Variablen und entspricht somit dem Slicer Exekutionsmodell.

Die Logic Klasse
Die Logic Klasse beinhaltet nur eine wichtige Funktion. „$\ddot{A}pply()$" wird aufgerufen wenn der Benutzer im GUI den Button 'Apply to whole Data' drückt. Die Logic Klasse führt darauf hin die ITK Pipeline aus, und stellt das Ergebnisbild im Slicer Planaransichtsfenster dar.

3.4.3 Slicer Support

Slicer bietet zwei Mailinglists, eine für Benutzer (slicer-users @ bwh.harvard.edu) und eine für Entwickler (slicer-devel @ bwh.harvard.edu) an. Diesen beiden Mailing Lists kann jeder beitreten der sich registriert. Hier werden Fragen zu unterschiedlichsten Themengebieten gestellt, dabei war der Eindruck, dass es sich um eine besonders hilfsbereite Community handelt, die offen für Probleme ist. Die beiden Mailing Lists enthalten ausserdem Archive, die man nach älteren Beiträgen für Hilfestellungen durchforsten kann. Für einen schnellen Start mit Slicer sorgen ausserdem Trainingsseiten, auf denen im Kurzdurchlauf einige der wichtigeren Module angeleitet erklärt werden.

3.4.4 Zukunftsaussichten & Abschließende Bemerkungen

Slicer3 bietet ein umfassendes Repertoire an Segmentations und Registrationsmethoden, eingebettet in ein vielfältig erweiterbares Framework. Die Möglichkeit verschiedene Klassen von Modulen in drei verschiednen Programmiersprachen(TCL, C/C++, Phyton) zu entwickeln, und die Verfügbarkeit auf Windows, Mac, und Linux und Solaris Systemen machen Slicer3 sehr flexibel. Nicht zuletzt profitiert davon eine immer größer werdende Community aus Programmierern und Benutzern, die sich in gegenseiteiger Absprache darum bemühen die Funktionalitäten von Slicer3 zu verbessern und zu erweitern. Wenn man bedenkt, dass es sich hierbei um eine Open Source Anwendnung handelt, deren Entwicklungsteam nichts dafür bezahlt bekommt, an dem Projekt weiterzuarbeiten, ist der Enthusiasmus mit dem die Leute ans Werk gehen, und mit dem sie bemüht sind bei allfälligen Problemen zu helfen, beeindruckend. Als Open Source Anwednung besitzt Slicer3 auch keine Freigabe der

Federal Drug Administration (FDA), und die Verwendung von Slicer erfolgt daher auf eigenes Risiko. Die Unterstützung durch bekannte Frameworks wie ITK, oder VTK ermöglicht es Programmierern mit entsprechendem Vorwissen, sich mit minimalem Arbeitsaufwand in die Slicer3 Architektur einzulesen und funktionsfähige Module zu schreiben. Zusätzlich sind die ITK , VTK Frameworks vielfach erprobt und haben sich in Performance und Rechenzeitaufwand bewährt. Für ungeübtere Programmierer ist die Implementation eines einfachen Moduls durch die Verwendung eines CLP Programmes, bzw. eines gescripteten Programmes schnell durchführbar. Ein negativer Aspekt macht sich bei der Verwendung von loadable Modulen bemerkbar, da diese nur dann verwendet werden können, wenn die Slicer3 Version die sie startet mit demselben Kompilieren kompiliert wurde wie das Plugin selbst. Eine Importation eines solchen Moduls in eine neue Version setzt daher immer vorraus, dass auf diesem Rechner Slicer3 und mit den gleichen Einstellungen kompiliert wird wie auf dem, auf dem das Plugin kompiliert wurde. Positiv wiederum ist, dass Plugins bei der Slicer3 Community eingereicht werden können. Wenn diese gewisse Vorraussetzungen erfüllen, werden sie in neue Slicer3 Versionen übernommen, und stehen dann in der vorkompilierten Version zu Verfügung. Ein Vergleich der Binary Versionen von Slicer 3.5 zu 3.6 zeigte, dass gut 10 neue Module neu integriert wurden. Wieder ein Beweis für die aktive Community von Slicer3. Wohin sich Slicer3 entwickeln wird abzuwarten sein. Auf jeden Fall wird Slicer 3.6 die letzte Version in der Slicer 3x Serie sein, die nächste Version heißt dann Slicer4 und soll vorraussichtlich im Jänner 2012 herauskommen. Derzeitige Bemühungen gehen unter anderem in die Richtung die GUI Komponente von KWWidgets auf Qt umzustellen. Qt ist neuer und besser auf unterschiedlichen Plattformen ethabliert als KWWidgets. Diese Veränderung wird bedingen, dass die selbst geschrieben GUI von loadable Modulen neu kreiert werden muss! CLP Module setzen ja auf ein XML um die GUI zu basteln, für sie wird sich, dank eines geeigneten Wrappers, nichts ändern. Des weiteren planen die neuen Änderungen Phyton mehr in den Vordergrund zu rücken und die TCL Funktionalität einzuschränken.

3.5 OsiriX

3.5.1 Einleitung

Osirix [7] ist eine frei erhältliche Bildverarbeitungssoftware / Picture Archivating System (PACS) Workstation, die auf die Verarbeitung von Dicom Files spezialisiert ist, aber auch viele andere Formate bearbeiten kann. In Osirix wurden große Teile des Dicom Standard für den Informations und Datenaustausch medizinischer Bilddaten implmenetiert, was es Osirix ermöglicht, mit anderen PACS) Systemen zu kommunizieren, und Daten von verschiedenen Stellen auch über das Internet zu beziehen, bzw. als Server zu fungieren, und Daten zu verschicken. Osirix ist auf die Verarbeitung multidimensionaler und multimodaler Daten spezialisiert: beginnend von der Darstellung von 2D Daten, ist eine 3D Rekonstruktion möglich. Zusätzlich kann eine 4te Dimension hinzugezogen werden (Zeitkomponente, z.B: beim Cardiac-CT), bzw. eine 5te (z.B: Zeitliche CT Aufnahme mit PET Daten) Komponente. Für die 3D & 4D Darstellung der Daten stehen Methoden, wie z.B: Surface Rendering, Volume Rendering und Maximum Intensity Projection (MIP) zu Verfügung. Die Visualisierungsmethoden basieren auf OpenGL Technologie, wie sie auch in den meisten Videospielen zum Einsatz kommen. Die Darstellung ist dadurch nicht nur extrem schnell, sondern basiert auch auf einem fundierten Standard, was die Erweiterbarkeit von Osirix verbessert. Osirix ist als Freeware nur in der 32 Bit, Version erhältlich. Die 64 Bit Version wird auf der Osirix Homepage kostenpflichtig vertrieben. In jedem Fall ist OsiriX ausschließlich für Apple Mac Systeme verfügbar. Als frei erhältliche Software wurde Osirix under der GLP (GNU Public Licence) veröffentlicht. Das ermöglicht es den Sourcecode von Osirix frei zu verwenden und zu verändern, und bietet somit auch die Möglichkeit, Plugins selbst zu schreiben um die Funktionalität von Osirix zu erweitern. Osirix besitzt eine große Community, und wird weltweit in mehreren Kliniken, Instituten, Hochschulen und Laboratorien für Lehr / Forschungszwecke eingesetzt. Nach Schätzungen der Entwickler arbeiten bis zu 50.000 Leuten mit Osirix. Osirix besitzt als Freeware in den USA und Europa keine Zulassung als Medizinprodukt und ist somit nicht für die Verwendung in der Therapie & Diagnose am Menschen zugelassen. Es wird allerdings von der Firma pixmeo [8] eine Version von Oirix (Osirix MD) vertrieben , die in den USA eine FDA und in Europa eine CE-1 Kennzeichnug besitzt, und für den Einsatz in der Befundung und Diagnose zugelassen ist. Osirix wird neben der Version für den Apple Mac auch als kostenfreie iPad/Iphone Version (Osirix HD) ebenfalls durch die Firma pixmeo angeboten.

[7] OsiriX - http://www.osirix-viewer.com
[8] pixmeo - http://pixmeo.pixmeo.com/

Schließlich gibt es noch (Osirix-Lite), eine CD Rom Version von OsiriX, bei der die Applikation ohne Installation von der CD Rom gestartet werden kann. Im Verlauf dieser Arbeit wurde nur die 32 Bit MacOSX Version von OsiriX untersucht.

3.5.2 Installation

Bei der Installation von Osirix kann man entweder auf eine vorkompilierte Version zurückgreifen, oder OsiriX selbst aus dem Sourcecode kompilieren.

Die vorkompilierte Version

Die derzeitige 32 Bit Freeware Version von Osirix (3.9) ist auf der OsiriX Homepage frei verfügbar, und kann ohne Vorinstallation diverser Programme installiert und verwendet werden. Die Systemvoraussetzungen sind:

- MacOS X 10.5 oder darüber.
- Intel Processor.
- Mind. 1 GB Arbeitsspeicher.

Je mehr Bilder in Osirix geladen werden müssen, desto stärker wirkt sich dies auf den verwendeten Arbeitsspeicher aus. Empfehlungen hierfür sind:

- 6 GB RAM bei mehr als 800 Bildern. (CT & MRI, PET-CT).
- 8 GB RAM bei mehr als 1500 Bildern. (Nur bei OsiriX-64 Bit!).
- 12 GB RAM bei mehr als 3000 Bildern. (Nur bei OsiriX-64 Bit!).

Wie man sehen kann ist die Verwendung der 32 Bit Version in der Anzahl der ladbaren Bilder beschränkt: 32 Bit Applikationen können nur bis maximal 4 Gb Arbeitsspeicher Bilder laden (die 2 zusätzlichen Gb bei 800 Bildern sind als Sicherheitsbuffer gedacht, damit der Mac nicht abstürtzt). Für alle anderen Anwendungen muss die allerdings kostenpflichtige 64 Bit Version von Osirix verwendet werden.

Die selbstkompilierte Version

Als Alternative zur vorkompilierten Version kann man Osirix selbst kompilieren. Will man Plugins entwickeln, muss man auf diese Option zurückgreifen, da Plugins bei der Kompilierung bestimmte Referenzfiles aus dem Sourcecode benötigen. Als ersten Schritt sollte man sich vergewissern, dass folgende Bedingungen erfüllt sind:

- mindestens Mac OS X 10.5 Leopard (10.6 oder 10.7 empfohlen).
- X-Code 3.2.
- Subversion.

X-Code ist Apples Entwicklunsgumgebung und ist, eine Registration auf der Apple Homepage vorrausgesetzt, gratis verfügbar. Subversion ist eine Software zur Softwareversions Verwaltung und ebenfalls gratis verfügbar. Hat man obige Bedingungen erfüllt, kann der OsiriX Sourcecode über die Shelleingabe

```
svn co https://osirix.svn.sourceforge.net/svnroot/osirix osirix
```

bezogen werden. Nachdem der Sourcecode heruntergeladen wurde, führt man die X-Code Projektdatei 'Osirix.xcodeproj' aus. Unter 'Targets' wählt man 'Unzip Binaries', woraufhin alle verwendeten Frameworks entpackt werden. Die Frameworks werden über obigen svn Befehl mit dem Sourcecode heruntergeladen, und liegen als statische Bibliotheken vor. Dies hat zum Vorteil, dass die einzelnen Frameworks nicht mehr kompiliert werden müssen, allerdings zum Nachteil, dass sie nur in 32 Bit kompilierter Version vorliegen. Wollte man OsiriX in 64 Bit kompilieren, müsste man sicherstellen, dass man eine ebenfalls 64 Bit kompilierte Version aller Frameworks verwendet. Informationen darüber, wie genau dies zu tun ist, ist sogut wie nicht vorhanden. Es gibt keine offiziellen Hilfestellungen auf der OsiriX Homepage, und auf ein diesbezügliches Mail an die OsiriX Newsgoup wurde nicht beantwortet, vermutlich eben deshalb weil die 64 Bit version kommerziell vertrieben wird. Ein weiterer Nachteil der statischen Bibliotheken ist, dass sie teilweise nur abgespeckte Versionen ihrer Originale sind, und nicht die komplette Funktionalität des gesamten Frameworks enthalten. Zudem können diese statischen Bibliotheken in manchen Osirix Versionen teilweise schon veraltet sein. (z.B Osirix 3.8 enthält VTK 5.0.x, die aktuelle Version von VTK ist zum Zeitpunkt dieser Arbeit allerdings 5.6.x)

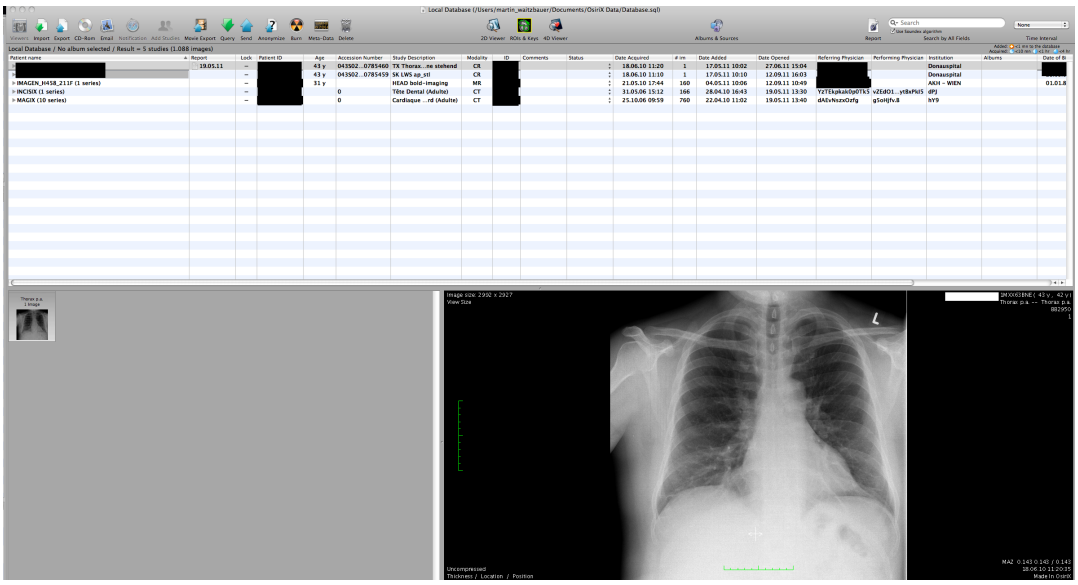

Abbildung 3.20: OsiriX Hauptbildschirm

3.5.3 OsiriX - Übersicht

Nach dem Start von OsiriX öffnet sich das Hauptfenster. Dies stellt eine Ansicht auf die gewählte Datenbank von OsiriX, mit all ihren Einträgen dar. Mit einem Doppelklick auf einen Datensatz öffnet sich dieser, und es öffnet sich das Datenansichtsfenster (Abb. 3.20). Im oberen Bereich befindet sich eine Ansicht auf die OsiriX Datenbank mit ihren Einträgen, rechts unten erhält man eine Vorschau auf den gerade angewählten Datensatz.

In der Hauptmenüleiste finden sich folgende Optionen:

OsiriX Preferences: Generelle Einstellungen zum Programm, Einstellungen zu: Netzwerk (Senden & Empfangen von Daten), zur Datenbank, Tastaturbefehle, Ansichtsfenster und Bildschirm Annotationen.

File: Beinhaltet Optionen zum Import und Export (Dicom, Quicktime, JPEG, TIFF, iPhoto) von Bildern. Brennt Daten auf CD, Anonymisiert Dicom's, und beinhaltet einige Funktionen zur Datenbank. Osirix ist in der Lage Dicom Files verschiedener Modalitäten (Röntgen, Ct, Mri, Ultraschall, Pet, Spect...) als Einzelfiles, Serien, oder aus DICOMDIRS zu laden. Dicom Bilddateien können auch geschrieben werden. Entweder aus anderen Dicom Files, um Duplikate zu erzeugen, das Dicom zu annonymisieren, oder als Secondary Capture Dicoms

(SC Dicom). SC Dicoms sind aus anderen Bildateien konvertierte Dicom's. Solange keine Konvertierungsmöglichkeit aus der Quelldatei direkt besteht (über metafiles, xml), müssen Dicom relevante Attribute händisch eingegeben werden. SC Dicom's beinhalten allerdings keine Informationen über Skalierung, oder die verwendete Modalität, und sind somit nur als rein visuelle Bilddaten zu verstehen. OsiriX kann Dicom Dateien mit folgenden Bildeigenschaften lesen:

- mono-frame, multi-frames.
  multi frame DICOMS werden als Serie geladen.
- Monochromatisch, RGB, YBR.
- Nicht quadtratisches Pixelseitenverhältnis (Pixel Aspect Ratio).
- Dicom's mit 8, 12, 16, 32 Bits
  Dies ist die Quantisierung des Bildes. Es lassen sich 256,4096,65536,4294967296 Graustufen darstellen (Abhängig von der Quantisierungsgenauigkeit des Monitors!)

Alle zusätzlichen Metainformationen, die in dem Dicom gespeichert sind, können in Osirix als Annotation angezeigt werden.

Zusätzlich zum Dicom Datenformat lassen sich noch folgende andere Formate laden:

- LSM Dateien von Zeiss (8, 16, 32 Bits) (Confokal Mikroskopie).
- BioRadPIC Dateien (8, 16, 32 Bits) (Confokal Mikroskopie).
- TIFF (8, 12, 16, 32 Bits), multi-pages.
- ANALYZE (8, 12, 16, 32 Bits).
- PNG, JPEG, PDF, Quicktime, AVI, MPEG, MPEG4.
- JPEG (verlustbehaftet + verlustfrei), JPEG2000.
- Papyrus Files.

und folgende Datein aus Dicom Dateien schreiben:

- TIFF
- JPEG
- Quicktime
- RAW

- PACS

Network: Hier können Dateien verschickt und empfangen werden, wenn ein entsprechender PACS Server angegeben wird.

Edit: Kopiert Daten, macht Bearbeitungsschritte rückgängig.

Format: Lässt den Benutzer die Menüleisten nach seinen Wünschen umgestalten.

2D Viewer: Einstellungen rund um das Bearbeiten von 2D Daten. Farbeinstellungen, Farbkorrektur, Farbwechsel. Es können auch Einstellungen am Viewer vorgenommen werden: Splitscreen, Verdoppelung und ob Farb, Positions und Lookuptable Einstellungen zwischen verschiedenen Viewern übernommen werden sollen. Weiters können Einstellungen zur Grauwertensterung vorgenommen werden. Hier finden sich auch einige Fensterungs Voreinstellungen für CT Aufnahmen für Knochen, Abdomen und Hirn. In dieser Rubrik befinden sich auch simple Bildfilter, wie z.B: Gausssche Glättungsfilter, Laplacefilter, sowie Hochpassfilter mit verschieden großen Kernels.(Berechnung im Ortsraum)

3D Viewer: Einstellungen rund um die Multidimensionale Darstellung. Multiplanare Rekonstruktion, Volume Rendering, Surface Rendering, 3D Endoscopy, mit jeweils unterschiedlich wählbaren Optionen.

ROI: Einstellungen für 2D / 3D ROI's. ROI's können entweder importiert werden, oder durch einen einfachen Region Growing Algorithmus selbst erstellt werden, die dann als File abgespeichert werden können. ROI's können nachbearbeitet werden (Erosion, dilatation, opening, closing).

Plugin: Hier werden alle Plugins aufgerufen, die lokal für OsiriX zur Verfügung stehen. Unter 'Plugins Manager' kann aus einer Liste von bestehenden Plugins ausgewählt werden, die von einem zentralen Plugin Server zur Verfügung gestellt werden. Zu jedem der Plugins gibt es meist eine kleine Beschreibung wie das Tool zu handhaben ist, und was es genau tut. Die Verwendung von heruntergeladenen Plugins erfolgt dabei auf eigene Gefahr.

Windows & Help: Fenstereinstellungen und Hilfeassistenten.

3.5.4 Plugins

Plugins sind in OsiriX das, was in Slicer3D Module sind. Plugins sind kleine eigenständige Programmelemente, die in OsiriX eingebunden werden, und über den Menüleintrag 'Plugin' gestartet werden. Plugins sind betimmten Kategorien zugeordnet:

- Image Filters: Umfasst Plugins, die Bilddaten modifizieren.
- ROI Tools: Plugins, mit denen Region of Interest (ROI's) erzeugt werden können. ROI's werden üblicherweise benutzt, um wichtige Stellen im Bild zu kennzeichnen. ROI's können manuell oder automatisch (z.B: durch einen Segmentationsalgorithmus) erzeugt werden.
- Database: Plugins, die mit OsiriX als Pacs Workstation zu tun haben.

Alle anderen Plugins werden 'Other' zugeordnet. Jede Osirix Version, egal ob sie selbst kompiliert oder als vorkompiliertes Package heruntergeladen wurde, besitzt eine gewisse Anzahl vorinstallierter Basis Plugins. Dazu zählen:

- ROI-Enhancement
- ResampleData
- ReduceSeries
- FillGaps
- Cobb Angle
- ExportROIs
- Normalize
- Ejection Fraction
- Invert Data
- OrthogonalReslice
- Ratio T2 Map
- Duplicate

- T2 Fit Map

Neue Plugins

Für das Einbinden eines neuen Plugins in OsiriX gibt es zwei Möglichkeiten:

1. Herunterladen:
   Über $Plugins \rightarrow Plugin\ Manager$ hat man die Möglichkeit eine Verbindung mit einem zentralen Pluginserver herzustellen, auf dem alle derzeitig für OsiriX entwickelten Plugins zur Verfügung stehen. Neben einem Beschreibungstext, liegt hier meist auch eine How-to Anweisung vor, die dem Benutzer mitteilt, wie das Plugin zu verwenden ist. Hat man das Plugin heruntergeladen, installiert es sich von alleine, und ist ab dann einsatzbereit.
2. Selber Programmieren:
   Plugins können in OiriX selbst geschrieben werden und müssen dann installiert werden. Eine einfache Methode das Grundgerüst für ein Plugin zu erzeugen ist der im Sourcecode enthaltene Plugin Generator. Dieser erzeugt eine neue Xcode Projektdatei, in der alle notwendigen Einstellungen vorgenommen wurden, damit die ausführbare kompilierte Datei von OsiriX importiert werden kann. Das kompilierte Plugin besitzt die Dateiendung '.osirixplugin' , und lässt sich einfach durch einen Doppelklick installieren.

Plugins in OsiriX werden mit einer Versionsinfo erstellt. Das ermöglicht es dem Benutzer den aktuellen Stand des Plugins zu überprüfen und im Falle einer neuen Version diese zu installieren. Sollte bereits ein Plugin einer früheren Version installiert sein, wird nach einer Bestätigung durch den Benutzer das alte Plugin durch das neue ersetzt.

Anmerkung: Das 'Installieren' eines Plugins entspricht in OsiriX in Wahrheit einer Verschiebung der ausführbaren Datei in den OsiriX Arbeitsordner $ApplicationData/Osirix/Library...$
Alle heruntergeladenen und selbst geschriebenen Plugins, also alle Plugins die nicht mit der Installation von OsiriX mitgeliefert wurden, befinden sich hier.

3.6 Programmieren in OsiriX

3.6.1 Systemarchitektur

Osirix ist aus mehreren plattformunabhängigen Frameworks aufgebaut (Abb. 3.21). Kitwares VisualisationToolkit VTK übernimmt die Aufgabe Grafiken darzustellen, und zu rendern. Da die VTK Renderroutinen auf OpenGL basieren, können sie extrem schnell durchgführt werden. Das Insight Tool Kit, ITK ist für allerlei Bildmanipulationsalgorithmen, (Filter, Segmentation, Registration) in OsiriX verantwortlich. Das DicomToolkit DCMTK [9] ist eine freie Bibliothek, geschrieben in C /C++, die den Dicom Standard weitgehend implementiert. Mit DCMTK können Dicom Bilddaten gelesen, konvertiert und gespeichert werden. Zusätzlich gibt es eine Netzwerkkomponente, die hilft Dicom Dateien über ein Netzwerk zu verschicken, bzw. zu empfangen. Papyrus [10] ist ähnlich wie DCMTK für das Lesen und Schreiben von Dicom Files zu verwenden, zusätzlich lassen sich damit allerdings noch Papyrus Files lesen und schreiben. Alle oben angeführten Frameworks sind unter dem open-source Konzept frei verwend und weiterverwendbar, was es für Benutzer und Entwickler vereinfacht, neue Funktionalitäten in OsiriX einzubauen, bzw. schon bestehende funktionierende Applikationen leichter in OsiriX umzusetzen.

3.6.2 Pluginarchitektur

Jedes OsiriX Plugin ist eine Unterklasse der OsiriX Klasse 'PluginFilter'. Nach dem Start von Osirix wird für alle Plugins der Klasse 'PluginFilter' die Funktion $initPlugin()$ aufgerufen. Hier werden allfällige Initialisierungen und Speicherallokationen vorgenommen, damit OsiriX weiß, wie aufwändig es werden wird das Plugin auszuführen, und zusätzlichen Speicher anfordern kann. Wenn ein OsiriX Plugin gestartet wird, wird es über die Methode $-(long)filterImage : (NSString*)menuName$ aufgerufen. Hier wird die eigentliche Funktionalität des Plugins implementiert. FilterImage erwartet zwar als Rückgabewert einen Datentyp der Form 'long', dies ist allerdings eher wie der Rückgabwert von bekannten $main$ Funktionen zu verstehen (z.B: $intmain(intargc, char*argv[])$. Der Rückgabewert von FilterImage kann somit immer auf $return\ 0$ gesetzt werden, sollte aber auf keinen Fall vergessen werden, da OsiriX das Plugin sonst nicht laden kann! Das Programmieren eines neuen Plugins

[9] DCMTK - http://dicom.offis.de/dcmtk.php.de
[10] Papyrus - http://www.dim.hcuge.ch/papyrus/01_Papyrus_Presentation_EN.htm

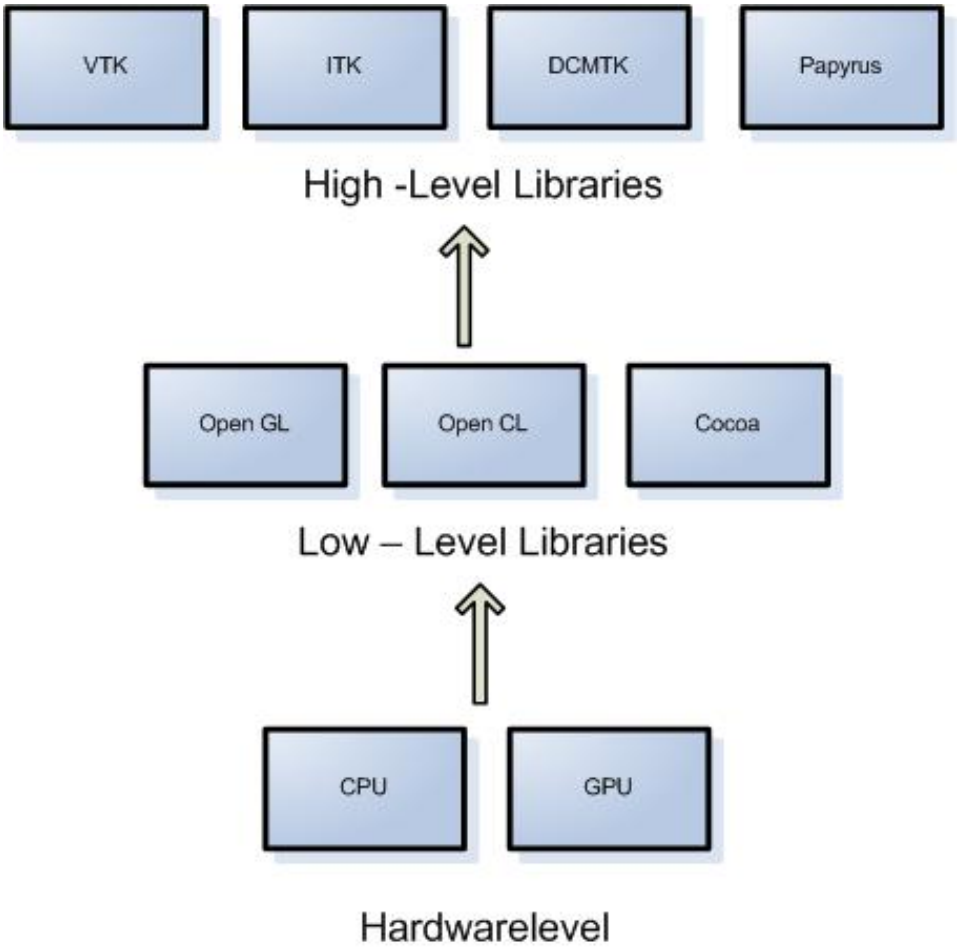

Abbildung 3.21: OsiriX Softwarearchtektur.

sollte am besten immer mit dem *Plugin Generator* im OsiriX Sourcecode Ordner begonnnen werden. Dieses Script erstellt selbstständig ein Xcode Projekt mit der Basiskonfiguration für OsiriX Plugins. Das neue Plugin Projekt befindet sich dann in dem Ordner, von dem der Plugin Generator gestartet wurde, und beinhaltet folgende Dateien (für ein HelloWorld Projekt):

- HelloWorl.xcoeproj: Die ausführbare X-Code Datei.
- HelloWorldFilter.h: Das Filter Header File.
- HelloWorldFilter.m: Die Implementation des Header Files.
- HelloWorld_Prefix.pch: Interne Projektdatei.
- Info.plist: Metainformationen zum Plugin.
- version.plist: Versionsinfo des Plugins.
- OsiriXHeaders: Beihaltet einige OsiriX Header Dateien.

Der Ordner 'Osrix-Headers' beinhaltet einige der wichtigsten grafischen und funktionalen OsiriX Elemente, die als Objective C Klassen implementiert sind und vom Benutzer frei verwendet werden können. Im Folgenden werden einige vorgestellt:

- Browser Controller:
  Ein Browser Controller (Abb. 3.22) beinhaltet eine grafische Tabellenansicht auf die OsiriX Datenbank. Hier können neue Bildserien geladen, bzw. im Previewbildschirm angezeigt werden. Wenn man einen Datensatz ausgesucht hat, öffnet sich ein Viewer Controller.
- Viewer Controller:
  Ein Viewer Controller (Abb. 3.23) zeigt die im Browser Controller ausgewählten Serien in einem seperaten Fenster an. Links sieht man die Kurzübersicht des ausgewählten Datensatzes, rechts eine Darstellung der Daten. Verschiedene Bilder können durch 'Blättern' mit dem Mausrad im Viewer Controller angezeigt werden. Ein ViewerController ist als übergeordnetes Objekt für DCMViews zu verstehen.
- DCMView:
  Eine DCMView (Abb. 3.24) ist für die Darstellung eines DCMPix Objektes zuständig. DCMViews enthalten Informationen zu Bildschirmannotationen, sowie eine Anzeige für Grauwerteinstellungen, welche durch horizontale/vertikale Bewegungen mit der Maus verändert werden können.

Abbildung 3.22: OsiriX BrowserController Element. [13]

- DCMPix:
  Die eigentlichen Bilddaten werden in OsiriX als DCMPix Objekte abgespeichert (Abb. 3.25). Diese sind über den Befehl $(NSMutableArray*) - pixList()$ der Klasse ViewerController anzusprechen In DCMPix werden Informationen über die Skalierung, Dimensionierung, den Datentyp, sowie Ursprungsangaben und Informationen zu ROI's festgehalten.
- dicomFile: Ein Objekt, das Informationen zu den geladenen Dicom Files enthält.
- ROI: Basisklasse für alle ROI's.
- MyPoint: Basisklasse für alle 2D Punkte.
- PluginFilter: Oberklasse des Plugins. Die wichtigste Methode hierbei ist $(NSArray*)viewerControllersList$, mit der auf die derzeit geöffneten ViewerController (und damit Bilddaten) zugegriffen werden kann.

Die oben vorgestellten Klassen sind jene, die nach der Erstellung eines Plugins mit dem 'Plugin Generator' im Xcode Projekt verfübar sind. Sollten andere OsiriX Klassen benötigt werden, müssen sie in den Header Ordner inkludiert werden. Um die verwendeten Frameworks ITK, VTK verwenden zu können, bedarf es einiger Modifikationen am Projekt. Eine genaue Beschreibung aller OsiriX Klassen findet sich in der OsiriX Online Dokumentation[11].

[11] OsiriX Online Dokumentation - http://osirix.svn.sourceforge.net/viewvc/osirix/Documentation/OsiriX/html/index.html

Abbildung 3.23: OsiriX ViewerController Element. [13]

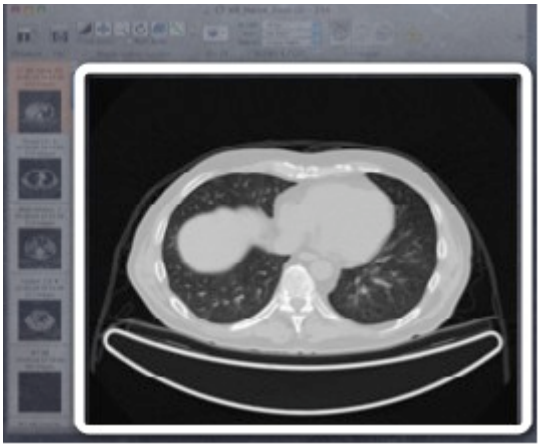

Abbildung 3.24: OsiriX DCMView Element. [13]

Abbildung 3.25: OsiriX DCMPix Element. [13]

3.6.3 OsiriX Frameworks einbinden

Um die Frameworks ITK und VTK in OsiriX Plugins verwenden zu können müssen kleine Veränderungen am Projekt vorgenommen werden:

ITK

Für die Verwendung von ITK ergeben sich zwei Möglichkeiten:

1. Man verwendet die statischen OsiriX ITK Bibliotheken.
2. Man verwendet eine selbstkompilierte Version von ITK.

Eine statische Version von ITK verwenden

Die erste Option ist in OsiriX einfach durchzuführen: ITK liegt in OsiriX als statische Bibliothek vor, und muss nur noch in das Projekt eingebunden werden. Dies geschieht indem man in Xcode die Projekteinstellungen unter $Targets \rightarrow Get\ Info$ aufruft. Unter 'User Header Search Paths' muss hier der Pfad zu den ITK Header Files angegeben werden. Die ITK Header Files befinden sich in $osirixsource/ITK180/Code$ in den Unterordnern:

/Algorithms /BasicFilters /Common /Numerics /IO /Numerics/FEM /Numerics/Statistics /SpatialObject /Utilities/MetaIO /Utilities/expat /Utilities/NrrdIO /Utilities/DICOMParser /Utilities /Utilities/vxl/vcl ITK180/Utilities/vxl/core.

Je nachdem welche ITK Klassen benötigt werden, muss der dazugehörige Ordner angegeben werden. Beim inkludieren der ITK Header Files empfiehlt es sich ausserdem, dies mit einer kleinen Modifikation zu machen. Da in Objective-C das Wort 'id' als Keyword für Datentypen jeglicher Art verwendet wird, in ITK/VTK 'id' jedoch als Variablenname, sollte man ITK/VTK Files nur mit Hilfe des Codestücks:

```
#define id iId
#import "itkImage.h"
#undef id
```

importieren. Dies definiert alle Variablen 'id' in den importierten Files als 'Id', und es kann zu keinen Fehlermeldungen durch Verwechslungen kommen. Nachdem die Header Files angegeben wurden, muss noch in 'User Library Paths' der Sourcecode Ordner eingetragen werden. Die vorkompilierten ITK Bibliotheken befinden sich in $osirixsource/ITK180/bin$. Diese müssen zusätzlich noch an zweiter Stelle im Projekt unter 'Frameworks & Librariers' angegeben werden. Dazu klickt man rechts auf das Ordnersymbol und fügt unter 'add existing files' alle mit *.a gekennzeichneten Bibliotheken aus dem $ITK180/bin$ Verzeichnis dem Projekt hinzu. ITK sollte nun vollständig in das Projekt inkludiert worden sein.

Eine selbstkompilierte Version von ITK verwenden

Dies ist schwieriger durchzuführen, als die statische Variante oben zu verwenden, dafür ermöglicht sie es, auf die gesamte Funktionalität des Frameworks zurückzugreifen, und zusätzlich selbst geschriebene ITK Klassen einzubinden. Die erste Schwierigkeit besteht darin, die selbstkompilierte Version von ITK so einzubinden, dass es zu keiner Verwirrung des Kompilers zwischen der statischen ITK Version von OsiriX und der selbst kompilierten ITK Version durch Duplikation der Klassen kommt. Dies entsteht dadurch, dass ITK standardmässig in dem Namespace 'itk' definiert ist. Eine einfache Methode dies zu umgehen, ist eine Refraktorierung des Namespace im ITK Sourcecode vorzunehmen. Glücklicherweise gibt es ein kleines Script, das von Brian Jensen geschrieben wurde [12]. Dieses Script nimmt genau obige Veränderungen am ITK Sourcecode vor, damit es zu keinen Verwechslungen mehr kommen kann. Sind alle ITK Klassen in den neuen Namespace umfraktoriert worden, muss ITK kompliert werden. Dabei sollte beachtet werden, dass ITK für dieselben Architekturen (32 Bit, 64 Bit) kompiliert werden muss, wie OsiriX, da der Klassenimport sonst Fehler verursacht. Für die Freeware Version von OsiriX muss hier die Architektur mit 32 Bit gewählt werden. Um ITK nun in ein OsiriX Projekt einzubinden, müssen die Projekteinstellungen unter $Targets \rightarrow Get\ Info$ aufgerufen werden. Unter 'User Header Search Paths' setzt man hier der Pfad zu den ITK Header Files der selbst kompilierten ITK Version. Für die ITK Binaries geschieht dies unter 'User Library Paths', die auf das 'bin' Verzeichnis verweisen sollten. Die ITK Bibliotheken (*.a) müssen unter 'Frameworks & Libraries' eingebunden werden. Hat man alle obigen Einstellungen vorgenommen, können nun alle ITK Header mit entsprechender Deklarierung im gewählten Namespace verwendet werden.

[12] Zur Zeit dieser Arbeit befand es sich auf http://osirixplugins.svn.sourceforge.net/viewvc/osirixplugins/NMSegmentation/refactor_

VTK

Die statischen OsiriX Bibliotheken von VTK befinden sich im OsiriX Hauptordner in den Verzeichnissen VTKHeaders bzw. VTKLibs. Um VTK zu inkludieren müssen zunächst die Archive (*.a files) in den Projekteinstellungen unter 'User Library Paths' und die Header Files unter 'User Header Paths' angegeben werden. Danach müssen die Header Files zusätzlich noch mit dem Projekt unter der Rubrik 'Frameworks' verlinkt werden.

Eine selbstkompilierte Version von VTK verwenden

Da OsiriX Frameworks nur in statischen Bibliotheken vorliegen, und standardmässig nicht von außen in das Projekt einbezogen werden, kann es passieren, dass OsiriX nicht immer auf dem neuesten Versionsstand seiner Frameworks steht. In so einem Fall bleibt nur noch die Möglichkeit die Funktionalitäten extern einzubinden. Bei der Verwendung von VTK entsteht das Problem, dass die Bibliotheken nicht in einem eigenen Namespace wie z.B: in ITK vorliegen. Wollte man nun eine eigene Version von VTK in das Projekt einbinden, käme es zu Fehlermeldungen des Kompilers, die aufgrund von Duplikationen derselben VTK Klasse herführen würden. Leider schlugen alle Versuche eine eigene Version von VTK einzubinden fehl, da es entweder wie oben beschrieben zu einer Fehlermeldung des Kompilers aufgrund von Namensduplikaten kam, oder es zu Fehlermeldungen im OsiriX Quellcode kam, bei dem Versuch das statische VTK Framework von OsiriX auszutauschen.

3.6.4 Umsetzung des MFP Moduls

Für die Implementation des Multy Frequency Algorithmus wurde (ebenso wie für Slicer3) die ITK Pipeline verwendet (Absch. 3.2.1). Der Ablauf entspricht in etwa demselben, wie dem des CLP Moduls (Abb.3.14). Es gibt jedoch einige Ausnahmen hinsichtlich des Ladens/Speicherns von Daten:

1. Bilddaten werden in OsiriX im Dateityp DCMPix * gespeichert, und liegen zur Laufzeit im lokalen Speicher vor. Der Datenimport zu einem ITK Image geschieht über einen itkImportImageFilter.

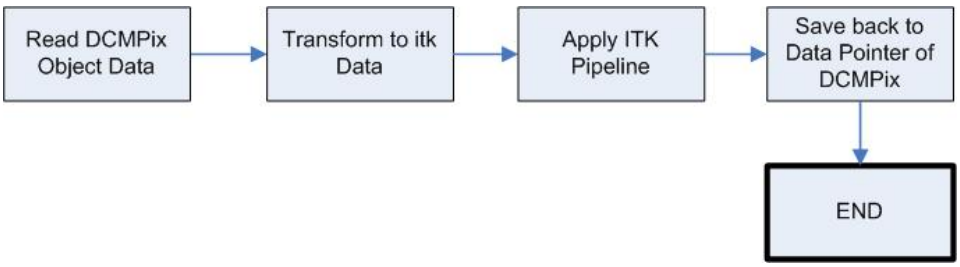

Abbildung 3.26: Diagramm des MFP Plugins in OsiriX.

Abbildung 3.27: Hauptbildschirm des MFP Plugins in OsiriX.

2. Bilddaten befinden sich immer im Dateityp float. OsiriX lädt standardmässig Bilddaten nur in diesem Format, was bedeutet, dass der Konvertierungsprozess von einem Datentyp in den von der ITK Pipeline verwendeten Datentyp float wegfällt.

3. Das Zurückschreiben der modifizierten Bilddaten geschieht über einen kleinen Trick: Anstatt die veränderten Daten in ein neues DCMPix zu schreiben, werden die Daten einfach mit der c Funktion $memcpy((void*)dest,(void*)source,longlength)$ in das float Array des alten Bildes geschrieben. Dies ist nicht nur um einiges schneller als ein neues Objekt anzulegen, es spart auch Speicherplatz.

Abb.3.26 gibt eine Diagrammübersicht über den Ablauf des OsiriX Plugins.

Das MFP Plugin befindet sich in der Menüleiste Plugins unter Filter. Beim Start des Plugins öffnet sich das Hauptfenster (Abb. 3.27).

Alle drei Fenster sind OsiriX Objekte vom Typ ViewerController. ViewerController 1 (links oben) zeigt die Originaldaten an, 2 (rechts oben) eine Kopie von ViewerController 1, der Modifikationen für den ganzen Datensatz anzeigt. 3 (links unten) ist ein Previewdatensatz, bestehend aus einem Originalbild der gewählten Schicht, allen Differenzbildern, sowie verbleibendem Hochpass und Summationsbild. Der Preview-ViewerController enthält außerdem eine Beschreibung des gerade betrachteten Bildes als Dicom Notationselement. Da alle Bilder des PreviewControllers als ein Datensatz betrachtet werden, gibt es für den ganzen Datensatz nur ein Grauwertfenster. Dies macht unter Umständen eine erneute Fensterung zwischen dem Betrachten der einzelnen Preview Schichten notwendig. Viewer Controller 2 und ViewerController 1 zeigen immer die selbe Datenschicht an (können aber andere Grauwert-Fenstereinstellungen verwenden). Das macht es einfacher spezielle Veränderungen mit den Originaldaten direkt zu vergleichen. Alle filterspezifischen Einstellungen werden an einem kleinen Fenster vorgenommen, das sich beim Öffnen des Plugins startet. (Abb. 3.28)

1. Einstellungen für die betrachtete Schicht des PreviewControllers. Bei Nicht-Schicht Aufnahmen hat dieses Feld keine Funktion. Sollte eine Schicht außerhalb des Datensatzes angewählt werden, wird die erste/letzte Schicht gewählt.
2. Einstellungsregler für die Gaussbänder. Der höchstmöglich einstellbare Wert beträgt 100. Da die Gaussfilter immer breitere Ausdehnung haben, man aber nur eine diskrete Einstellung mit den Reglern vornehmen kann, wurden die Einheiten auf den Reglern so umgerechnet, dass sie bei höchster Einstellung eine Gausskurve erzeugen, die an den Bildrändern noch 40% der Gesamtintensität aufweist. Das Maximum der der Differenzbänder der Gausskurven (entspricht den Filterkerneln) wird daneben in Linienpaaren / mm angegeben. Jedes der Bänder besitzt dabei eine einzelne farbliche Codierung, die Summe ist weiß, und der verbelibende Hohpass wird grau angezeigt
3. Die visuelle Darstellung der Gaussbänder. Diese Grafik zeigt einen Querschnitt durch das Filterelief der Gaussbänder. Die X Achse ist in Linienpaaren / mm angegeben.
4. Querschnitt durch das Powerspektrum der gerade betrachteten Schicht, mit Angabe der entsprechenden Linienpaaren / mm. Diese Ansicht gibt Aufschluss über die spektrale Verteilung der Frequenzen. Da sich die Werte von kleinen zu hohen Frequenzen um mehrere 10er Potenzen unterscheiden können handelt es sich hier um eine logarithmierte Darstellung.
5. Scaling Factors zu den gewählten Bändern. Die maximale Verstärkung beträgt 50.
6. Eine Anzeige über die gerade berechnete Schicht.

Das Plugin besteht aus 5 Klassen, wobei 4 für die grafische Darstellung der 2D Previewbänder und des Fourierspektrums sind. Beim Start des Plugins über die Funktion $-(long)filterImage:(NSString*)menuName()$ werden zunächst Initialisierungen der einzelnen ViewerController vorgenommen: ViewerController 2 aus Abb. 3.28 bezieht seine Daten von ViewerController 1, Viewer Controller3 richtet sich den Speicherplatz für alle Vorschaubilder und das Summationsbild ein. Danach wird die Initialisierung des GUI vorgenommen, sowie die gewählte Schicht fouriertransformiert, zentriert, und permanent im lokalen Speicher gehalten, damit die Verarbeitung der Preview Pipeline schneller geht. Danach erwartet das Programm Eingaben des Benutzers über die verschiedenen Regler. Abb. 3.29 zeigt das 'Use Case Diagramm' des Osirix Plugins. Im Wesentlichen unterscheidet es sich nicht von dem Slicer3 Plugin, mit der Ausnahme, dass der Benutzer hier 3 verschiedene Datensätze zur selben Zeit durchblättern/betrachten kann. Im Gegensatz zu Slicer3 gibt es in OsiriX keine MRML Knoten, somit werden auch keine zusätzlichen Objekte durch Benutzereingaben erstellt. Der Benutzer kann auch nicht mehr entscheiden welchen Input/Output Datensatz er verwenden will. Stattdessen wird der MFP Algorithmus auf den gerade geöffneten Datensatz angewendet. Auch bei OsiriX wurde die Anzahl der verwendeten Bänder auf 6 beschränkt, da die GUI statisch über den X-Tools Interface Builder kreiert wird. Für die grafische Darstellung von PreviewController3, und 4, wurde das Open Source Projekt 'Coreplot' [13] verwendet. CorePlot arbeitet mit dem auf Mac Systemen üblichen Datasource Konzept. Daher gibt es 2 Klassen für die Implementation des Gaussschen Wellenprofiles, eine Klasse die die Plots verwaltet, und eine Klasse die das Datasource Modell für Gaussdaten implementiert. Dieselbe Vorgehensweise gilt für die Darstellung des Fourierspektrums.

3.6.5 OsiriX Support

OsiriX Support findet sich zum einen durch Webstreams, und (kostenpflichtige) Handbücher[14]. Dieser Support bezieht sich allerdings nur auf Support für Benutzer. Ein weiterer Anlaufpunkt für Hilfesuchende wird über Yahoo Groups angeboten, wobei es eine frei zugängliche englischsprachige und eine französischsprachige Gruppe gibt. Im Verlauf dieser Arbeit wurde nur die englischsprachige Gruppe konsultiert, und in Summe waren die Eindrücke eher negativ. Es gibt zwar viele Personen die täglich neue Nachrichten schreiben, aber von den über 5000 (Stand Mitte Mai 2011) registrierten Mitgliedern kommt offenbar nur von einer Handvoll eine Antwort. Viele der gestellten Fragen richten sich außerdem nur

[13] Core Plot - http://code.google.com/p/core-plot/
[14] OsiriX handbücher - http://www.osirix-viewer.com/Learning.html

Abbildung 3.28: Das MFP Plugin Fenster in OsiriX

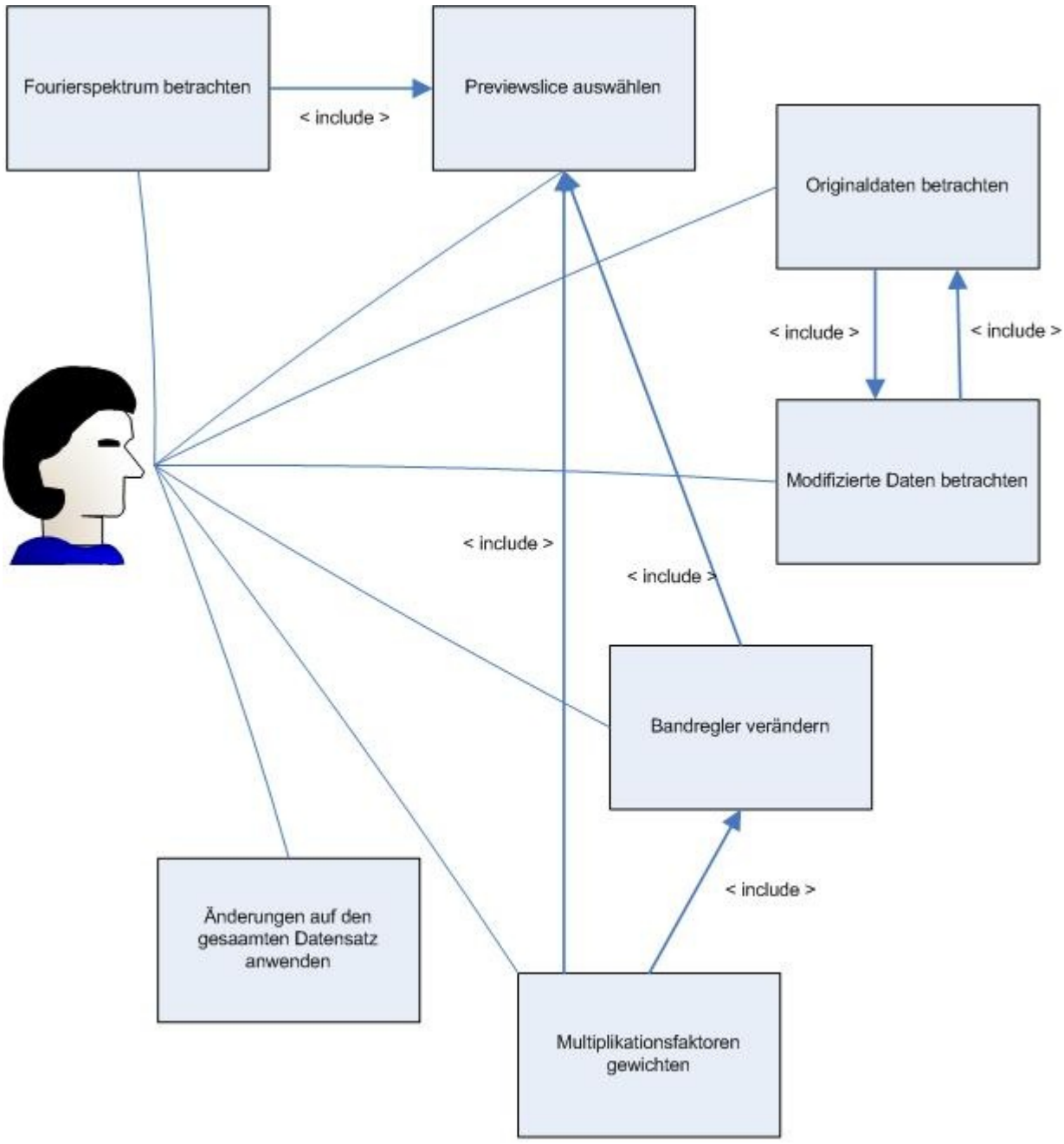

Abbildung 3.29: OsiriX MFP Plugin Use Case Diagram

an die Funktionalitäten des OsiriX Viewers, werden also von User Seite gestellt. Auf diese Fragen kann man häufigr mit einer Antort rechnen, als bei speziellen Fragen zur Programmierung in OsiriX. In so einem Fall ist der beste Support den ein Entwickler erhalten kann der Online Osirix Support. Neben einer ausreichenden Dokumentation des Sourcecodes finden sich hier allerhand hilfreiche Links. Eine weitere gute Hilfestellung bietet OsiriX Plugin Basics [15]. Dies ist eine Seite wo Entwickler Beispielplugins erklären, was es zu Beginn vereinfacht, selbst zu entwickeln. Diese Seite ist offiziell nicht auf der OsiriX Homepage verlinkt, enthält aber eigentlich die besten Informationen was das Schreiben von Plugins anbelangt. In Summe ist der OsiriX Support recht dürftig, wobei Benutzer des Viewers hier noch besser umsorgt werde,n als Plugin Entwickler.

3.6.6 Zukunftsaussichten und abschließende Bemerkungen

OsiriX ist ein sehr erfolgreiches Open Source Projekt, das sich seit seiner Beginnphase 2003 ständig weiterentwickelt hat. Inzwischen wird eine OsiriX Version (Osirix HD) für iPad, eine für die Befundung zugelassene (Osirix MD) und eine kostenpflichtige 64 Bit Version angeboten. Ursprünglich als Open Source Projekt gedacht, zeichnet sich hier ein gewisser Trend in Richtung Vermarktung von OsiriX ab. Die letzten großen Ankündigungen auf der OsiriX News Page kündigten genau nur jene oben genannten Versionen an. Offenbar hat die Industrie erkannt, dass OsiriX weltweit auf große Akzeptanz im medizinischen Umfeld stößt, und versucht nun Gewinn aus dem Produkt zu schöpfen. Als Indiz dafür könnte man den oben beschrieben schlechten Support für OsiriX sehen: Wenn man sich nicht das OsiriX Benutzerhandbuch kauft, steht man mit vielen Problemen allein da, denn Hilfe über die von der auf der offiziellen Homepage angegeben Newsgroup kommt selten. Sollte, wie angenommen die Vermarktung von OsiriX immer weiter in den Vordergrund rücken, so muss vermutlich auch damit gerechnet werden, dass hinsichtlich Performance und Funktionalität, zwei Linien gefahren werden. Im Moment zeichnet sich zwar noch nicht ab, ob es inhaltliche Unterschiede zwischen Freeware und kostenpflichtigen OsiriX Versionen geben wird aber mit einem schlechten Support von Amateur Entwicklern/Benutzern für die Freeware Version und einem Firmensupport für die kostenpflichtige Version, wird wohl auf lange Sicht die Bedeutung der kostenpflichtigen Version zunehmen. Was dies dann für OsiriX als Open Source Projekt für die Zukunft bedeutet wird abzuwarten sein.

[15] OsiriX Plugin Basics - http://osirixpluginbasics.wordpress.com/

3.7 Analyze 10.0

3.7.1 Einleitung

Analyze [11] ist eine von der Firma Mayo Clinic entwickelte Software zur Verabeitung, Registration, und Segmentation von medizinischen Bilddaten unterschiedlicher Modalitäten und Dimensionalitäten. Analyze 1.0 wurde anfangs der 70er Jahre für UNIX entwickelt, ist seitdem aber in mehreren Nachfolgeversionen erweitert worden. Die aktuelle Version 10 ist auf Windows, Linux, Mac und Solaris (32 / 64 Bit) gleichermaßen verfügbar. Der Großteil der Analyze Rechenalgorithmen basiert auf den über 600 Funktionen seines Verarbeitungsframeworks AVW, das ebenfalls Produkt der Firma Mayo Clinic ist. Neben AVW wird zusätzlich Insights ITK Softwarekit verwendet, das DicomFramework DCMtk, sowie das TCL basierte TCL/TK [16] zur Generierung der GUI Komponenten von Analyze. Analyze und AVW (sowie einige Analyze Module) sind lizenzpflichtige Produkte. Die Lizenz ist über das Mayo Clinic Sales Department [17] zu beziehen. Dabei kann man sich zwischen einer floating mode Lizenz, bei der ein Server eine Verwendungslizenz für einen Computer bereitstellt, und einer node - locked Lizenz, bei der eine Lizenz fix an die Mac Adresse eines Computers gebunden ist, entscheiden. Analyze ist rein für die klinische Forschung ausgelegt, und ist nicht als Befundungssoftware zugelassen.

3.7.2 Installation

Die Installation ist entweder über die mitgelieferte CD möglich oder über die Analyze Direct Homepage herunterzuladen. In jedem Fall ist Analyze schon so konstruiert, dass alle Frameworks vorkompiliert in Archiven vorliegen, und bei der Installation entpackt werden. Die Installation selbst wird über einen Wizard angeleitet, und kann in verschieden umfangreichen Ausmaßen durchgeführt werden.

3.7.3 Analyze - Übersicht

Beim Start von Analyze öffnet sich das Hauptfenster von Analyze, gennant 'Workspace' (Abb. 3.30). Von hier werden alle weiteren Funktionen, Laden, Speichern oder Bearbeitsungspro-

[16] TCL/TK - http://www.tcl.tk/software/tcltk
[17] Mayo Clinic Sales Department - http://www.analyzedirect.com/buy/

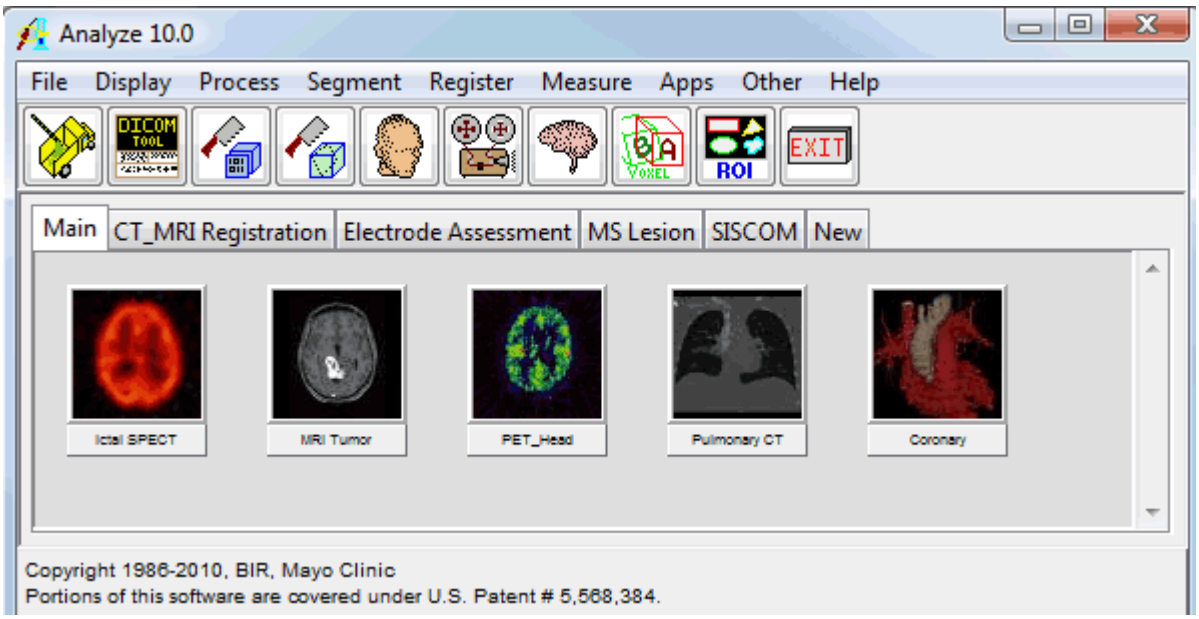

Abbildung 3.30: Analyze Workspace

zesse gestartet, und hier werden auch alle geöffneten Datensätze angezeigt. Analyze führt diese Funktionen in kleinen seperaten Fenstern, genannt 'Add-ons' (vergleichbar mit Modulen von Slicer3 oder Plugins von OsiriX) aus. Dabei fällt dem Benutzer gleich auf, dass man eine durchgehende Konzeptguideline für Add-ons bei Analyze vergebens sucht: Add-ons sehen komplett unterschiedlich aus, besitzen teils gleiche Funktionen (z.B zum Rotieren), die aber durch jeweils seperate Menüs von unterschiedlichen Einträgen in der Menüleiste aufgerufen werden. Zusätzlich kann es vorkommen, daß Symbole unterschiedliche Bedeutung haben. Dies macht es anfangs recht schwer und unübersichtlich sich mit Analyze zurechtzufinden. Um so mehr, da sich durch jedes vom Workspace aufgerufene Add-on ein seperates Fenster öffnet. Add-on Fenster können sich gegenseitig überdecken und sind dabei auch unterschiedlich groß. Will man wieder den Workspace erreichen, muss man sich ersteinmal durch die verschiedenen offenen Fenster durchklicken. Der Analyze Workspace wird beim Start von Analyze als physisches Verzeichnis $C:/Temp$ bei Windows Systemen und $/tmp$ bei Linux/Unix/Mac Systemen) angelegt. Wenn Bilddaten geladen werden, wird eine Kopie in diesem Verzeichnis erstellt, und beim Beenden von Analyze wieder gelöscht. Für jeden Benutzer von Analyze wird ein Unterordner angelegt, womit jeder Benutzer seinen eigenen Workspace verwenden kann.

3.7.4 Die Basismodule

Im Folgenden werden die Basismodule von Analyze vogestellt, die sich alle über die Hauptmenüleiste des Analyze Workspace Fensters starten lassen. Analyze benutzt sogenannte 'App-default' Dateien, die Standardeinstellungen zu Analyze Modulen, und Analyze speichern. Es handelt sich hier um einfache Textfiles mit bestimmter Syntax, die von Analyze zum Systemstart aufgerufen werden, um Module mit denen darin angegeben Einstellungen zu laden.

File

Bilddaten laden:
Mit Analyze lassen sich eine große Anzahl verschiedener Bilddatenformate laden. Die Funktionalitäten für das Laden von Bildern nimmt Analyze aus der AVW Library. Bilddaten können auf zwei Arten geladen werden: 'Load' öffnet ein Bild normal, über den Eintrag 'Load As' können Bilder vor dem Öffnen vorbearbeitet werden. Es können etwa die Bilddimensionen oder die Pixeldimensionen verändert werden. Es ist weiters möglich nur einen Bildausschnitt des Bildes zu laden, oder einen Rahmen um das Bild zu ziehen (Padding). Für Bilddaten ohne AVW Standard Import Funktion, muss entweder eine entsprechende Library geladen werden, die die Funktion übernimmt, oder ein spezielles Importer Konfigurationsfile geschrieben werden. Eine Liste aller verfügbaren Bildformate und der dazugehörigen Bibliotheken findet sich unter $Tools \rightarrow External\ Libraries$. Speziell sollte darauf geachtet werden, dass eine Inaktivierung eines Datenformats unter Umständen eine Inaktivierung mehrere anderer Datenformate zu Folge hat. Deswegen gilt es immer genau auf die spezielle Bibliothek zu achten, die die Lade-Funktion übernimmt.

Dicom Tool:
Das Dicom Tool verwaltet eine Datenbank, die für Dicom Serien ausgelegt ist. Hier kann der Benutzer durch alle Schichten des Datensatzes bewegen und dabei die Grauwerteinstellungen verändern. Für die weitere Verarbeitung mit Analyze kann entweder eine Schicht einzeln geladen werden oder als Volumen importiert werden. Importierte Volumen erscheinen im Workspace von Analyze. Das Dicom Tool besitzt zusätzlich eine Netzwerkkomponente, mit der Dicom Bilddaten empfangen und verschickt werden können.

Import/Export:
Das Import/Export Modul dient einerseits dazu 2D Bildstapel, als 3 dimensionale VolumeFiles zu öffnen, andererseits hilft es beim Laden von unbekannten Bildformaten, oder bei der Konversion zwischen Bilddateien. Dabei wird davon ausgegangen, daß Bilddaten in korrekter anatomischer und lexikaler Folge aufeinander gespeichert sind. Ist dies nicht der Fall, kann ein Bilddatensatz vor Konvertierung in ein VolumeFile durch das Sorting Modul unter $Tools \rightarrow Sorting$ noch in die gewünschte Reihenfolge gebracht werden. Das Raw-Data Modul unter $Tool \rightarrow Raw\ Data$ kann Daten eines unbekannten Bildtyps ohne die Hilfe einer entsprechenden Bibliothek lesen, wenn Bildaten unkomprimiert und in in kontinuierlicher Reihenfolge gespeichert sind. (Zusammenhängender Bytestream der Bilddaten.) Der Benutzer muss hier einige zusätzliche Informationen über das Bild angeben, etwa die Pixeldimensionen und Bilddimensionen, sowie die Bitanzahl pro Pixel. Je nach Einstellungen kann ein und dasselbe Bild komplett anders aussehen (Abb.3.31).

Display

Multiplanar Sections & Oblique Sections

Mit diesen Modulen können beliebige Schnittführungen durch den Datensatz gemacht werden. Wahlweise kann hier eine 2D Ansicht, oder eine 3D Ansicht gewählt werden.

Volume Compare

Bietet eine Möglichkeit zwei Datensätze zu vergleichen. Mittels der 'Blend' Option kann der Benutzer unterschiedliche algebraische Operationen (z.B: Mittelwert oder Differenzbild) zweier Datensätze bilden, und sich diese anzeigen lassen. Das Ergebnisbild kann zusätzlich mit Prozentanteilen der Vergleichsbilder gewichtet werden.

Volume Render & Surface Render

Diese Module rendern eine 3D Ansicht von Datensätzen, Object Maps, oder Surface Maps, die vorher mit anderen Modulen erstellt wurden.

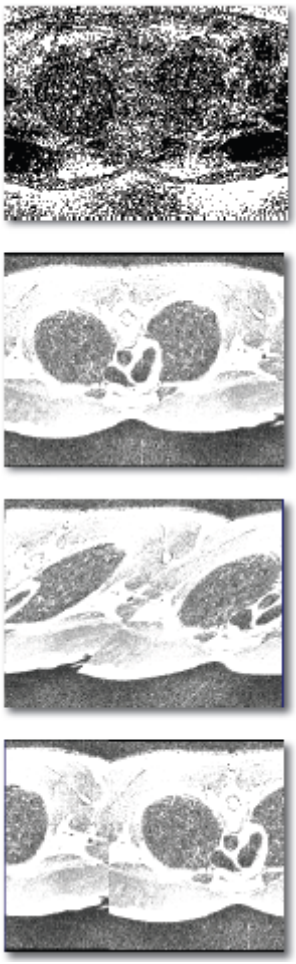

Abbildung 3.31: Dasselbe Bild mit untersheidlichen Einstellungen für (Von oben nach unten) Bittiefe, richtige Einstellungen, Falscher Urpsrung und falsche Bildbreite, Falscher Ursprung. [2]

Virtual Endoscopy

Mit diesem Modul lassen sich virtuelle Streifzüge aus der Egoperspektive durch eine beliebige Schnittebene des Datensatzes unternehmen. Der Benutzer kann frei die Richtung bestimmen, in die er sich begeben will, und bekommt ein in Echtzeit berechnetes, und gerendertes Bild. Virtuelle Endoskopie wird verwendet um baumartige Strukturen des Körpers (Luftröhre, Darm, Blutgefäße) zu untersuchen, wenn dies aus bestimmten Gründen auf herkömmliche Weise nicht möglich ist.

Movie

Modul zum Erstellen von Filmen aus einem 3D Datensatz.(Quasi 'Live' Mitschnitt).

Process

Module in der Rubrik Process verändern aktiv Pixelwerte des Originalbildes, z.B: durch algebraische Operationen oder Filter. Wenn ein Datensatz mit einem Modul aus dieser Rubrik erzeugt wurde, wird die modifizierte Kopie im Workspace abgelegt.

Image Calculator & Image Algebra

Zwei Module, die eigentlich das gleiche tun: Ein oder mehrere Datensätze mit vom Benutzer beliebig zusammengestellten Formeln berechnen. Formeln können in einem Taschenrechner eingegeben werden, und dienen als Filteroperationen für die Eingabebilder. Hier erstellte Bilder scheinen immer im Analyze Workspace auf.

Spatial Filters

Lässt den Benutzer aus einer Vielzahl von Bildfiltern auswählen, mit denen der Datensatz bearbeitet wird. Zusätzlich kann er sich entscheiden ob er die Filtermaske über den ganzen Datensatz legen will, oder nur Teilbereiche/Teilvolumen bearbeiten will.

Filter Designer

Ein Tool, mit dem händisch Filterübertragungsfunktionen im Fourierraum gezeichnet werden können, oder voreingestellte übernommen/verändert werden können (Gaussian, Butterworth). Dies gibt dem Benutzer die Möglichkeit verschiedene Filtermasken und ihre Wirksamkeit auf das Bild zu überprüfen. Das Ergebnisbild, die Filtermaske und die PSF sind einzeln als Bild aufruf- und speicherbar. Weiters können PSF's als Datentypen abgepeichert werden.

3D FFT

Die Anwendung der 2D FFT auf einen 3D Datensatz. Das Tool bietet die Möglichkeit PSF Filtermasken zu laden, und auf den Datensatz anzuwenden.

Segment

In der Rubrik Segment finden sich Module zur händischen und automatischen Segmentierung von Objekten. Segmentierte Objekte können klassifiziert werden, und bilden somit den Grundstein für andere Methoden wie automatisiertes Lernen oder Mustererkennung.

Image Edit

Hier kann der Benutzer händisch ROI's auf den Datensatz einzeichen, und diese als 'Object Maps' Dateien speichern. Halbautomatische Segmentierung ist zusätzlich über eine 'Auto Trace' Funktion möglich, die zusammenhängende ROI's anhand eines festgelegten Schwellwertes zeichnet.

Watershed Segemention

Segmentation mittels des Watershed Algorithmus.

Multispectral Classification (MSC)

Multi spektrale Klassifikation ist ein Segmentierungsalgorithmus, bei dem Informationen aus verschiedenen Spektren des Datensatzes genommen werden, um Objekte zu segmentieren. Vergleichbar ist dies etwa mit den 3 Kanälen (R,G,B) eines Farbbildes: Jedes dieser spektralen Bandbilder besitzt eigene Bildeigenschaften, die Objekte unterschiedlich zur Geltung bringen. Bei der MSC können z.B MRI Datensätze, mit T1,T2, und PD (Proton Densitity) Gewichtungen für eine Segmentation verwendet werden.

Object Extractor & Morphological Segmentation

Das Object Extraktor Modul implementiert automatische Objekt Erkennung mittels einer Verarbeitsungpipeline von Treshholding, Erosion, Dilatation und Region Growing. Objekt Extraktor benötigt einen Datensatz von mindestens 3 Schichten um zu funktionieren. Das erstellte Objekt kann darauf hin als Object Map Datei gespeichert werden, und kann als Input für andere Module dienen. Morphological Segmentation hat dieselbe Funktionsweise, wie der Object Extractor, nur muss hier alles händisch gemacht werden.

Surface Extraktor

Ein Modul zur Berechnung von Oberflächenmodellen aus einem 3D Datensatz. Surface Extraktor erstellt Surface Map Dateien, die mit anderen Modulen, wie z.B: dem Surface Render grafisch dargestellt werden können.

Register

Hier finden sich alle Funktionen, die mit 2D / 3D Registration zu tun haben, und in verschiedener Ausführung (rigid/nonrigid) verfügbar sind.

Measure

Measure Module bieten die Möglichkeit Statistiken aus Datensätzen zu gewinnen.

Line Profile

Dieses Modul gibt eine 2D Intensitätsverteilung einer vom Benutzer gezeichneten Linie durch den gewählten Schnitt im Datensatz.

Stereology & Region of Interest

Module für ROI's, mit dem der Benutzer ROI's erstellen, laden, verändern und speichern kann. Zusätzlich kann er sich statistische Informationen und Histogramme seiner gezeichneten ROI's anzeigen lassen.

Object Counting

Ein Modul, zum automatischen Zählen von Objekten im Datensatz. Objekte können entweder über eine Object map geladen werden, oder durch Segmentationsmethoden aus dem Datenstaz berechnet werden. Die Ergebnisse werden in einer Liste ausgegeben, die Grobcharakteristika jedes einzelnen Objektes festhalten (Ort, Größe, Min/Max/Durchschnitts Grauwert).

Tree Analysis

Ist ein Modul zur automatischen Segmentation von 'verzweigten Strukturen' z.B: Luftröhren, oder Blutgefäßystemen. Der Benutzer kann einen virtuellen Streifzug durch die Gefässsysteme unternehmen, und erhält dabei spezifische Informationen über gewisse Abschnitte. (Verzweigungswinkel, Durchmesser, Längenmessungen).

Apps

Apps beinhaltet Spezialmodule, die auf einen gewissen Anwendungskontext maßgeschneidert sind. Alle Module in dieser Sektion sind lizenzpflichtig, und müssen zusätzlich vom Sales Department von Mayo Clinic angefordert werden.

- SISCOM: Modul zur Früherkennung von epileptischen Anfällen.

- 3D Brain Atlas: Lerprogramm, mit dem anatomische und funktionale Einheiten des Gehirns in 3D beobachtet werden können.
- DTI: Ein Modul zur Berechnung und Visualisierung von Diffusions Tensoren.
- MRT2: Modul zur optimalen Kontrastdarstellung von T2 gewichteten MRI Bildern.
- VolumeMetrics: Modul zum quantitativen Vergleich der Ähnlichkeit zwischen zwei Datensätzen.

ITK

Diese Sektion bildet eine Referenz auf das Insight Toolkit ITK, das ebenso wie Analyze Methoden zur Segmentation, und Registration bietet.

Other

Bietet Informationen zum System, auf dem Analyze installiert ist, ein Vergrößerungsglas, sowie allgemeine Optionen zu Analyze.

3.7.5 Umsetzung des MFP Moduls

Analyze ist ein 'Stand alone' Produkt, das in sich geschlossen von der Firma Mayo Clinic vertrieben und gewartet wird. Ein Plugin für Analyze zu schreiben, das aus einer der Menüleisten von Analyze gestartet werden kann, ist nur möglich, wenn man die Lizenz für das 'Developper's Addon' [18] erwirbt. Weiters bietet der Support von Analyze an, auf Anfrage von Kunden maßgeschneiderte Module zu entwickeln, die nötigen finanziellen Mittel vorausgesetzt. Da während dem Verlauf dieser Arbeit beides nicht verfügbar war, wurde der MFP Algorithmus als Standalone Applikation mittels des GUI Toolkits KWWidgets, VTK, und der AVW Library durchgeführt. Das entstandene Programm ist eine einfache Viewing Software, die alle 2D Bilddaten laden kann, die von der AVW Library, und damit auch von Analyze unterstützt werden, nicht aber Bilddaten speichern kann. Vor dem Start des Programms muss sichergestellt werden, daß der entsprechende Mayo Clinic Lizenzserver für AVW läuft, da es sonst zu einem unerwarteten Absturz des Programms kommt! Beim Start des Programms muss der Benutzer über die Menüleiste $File \rightarrow Open$ einen Datensatz laden. Die Ansicht wechselt

[18] Näheres zum Developper's Addon unter http://www.analyzedirect.com/buy/

zu Registerkarte 1, wo das Originalbild angezeigt wird. Durch Verschieben der Regler links steuert der Benutzer die Standardabweichungen der Gaussfilter im Fourierraum. Die Ansichten aller aktiver Wellenfronten werden ihm farblich codiert in einem Fenster direkt über den Reglern angezeigt(grau = verbleibender Hochpass, schwarz = Summe aller Bänder). Bei der Anzeige dieser Grafik handelt es sich um dieselbe Klasse, die auch schon bei dem Slicer3 Modul eingesetzt wurde. Deswegen ist hier wieder die X Achse künstlich verlängert, ohne dass der eigentliche Datensatz sich soweit erstrecken würde. Unterhalb jedes der Regler, bis auf den ersten, wird zusätzlich angezeigt, wo sich das Maxmimum der entsprechenden Wellenkurve befindet. Die Angaben werden dabei in Linienpaaren / Millimeter gemacht. Diese Information stammt aus dem Tag 0028,0030 (PixelSpacing) des Dicom Header Files. Sollte diese Information nicht gefunden werden wird eine entsprechende Benachrichtigung ausgegeben. Jedes der einzelnen Bandfenster zeigt das entsprechende Subtraktionsbild an. Durch Pressen der linken Maustaste und Verschieben in X / Y Richtung lassen sich die Fensterungseinstellungen für Grauwerte verändern. Jedes der Subtraktionsbilder besitzt eine eigene Fensterung, da sonst die Unterschiede zwischen den einzelnen Bildern nicht gut erkennbar wären. In der letzten Registerkarte finden sich die Einstellungen zu den Multiplikationsfaktoren, sowie die entsprechende Ansicht des Summationsbildes. Abb. 3.32 zeigt ein genaues Aussehen.

3.7.6 Analyze Support

Analyze 10.0 ist ein kommerziell vertriebenes Produkt, jeglicher Support kommt aussschließlich vom Analyze Support Team. Auf der Analyze Homepage[19] finden sich neben einer gratis online Dokumentation kleine Movies, die bei einem Einstieg in Analyze helfen sollen. Des weiteren gibt es das 'Analyze Direct Maintenance Agreement', einen speziellen Supportservice für Kunden, die die neueste Version für Analyze gekauft haben. Das 'Maintennace Agreement' gilt für ein Jahr und beinhaltet unlimitierte kostenlose Telefon und E-mail Kommunikation mit dem Analyze Support Center, sowie der Möglichkeit, dass sich Mayo Clinic Mitarbeiter über einen Remote Link direkt mit dem Computer des Kunden verbinden, dort seine Maus steuern und ihm bei der Lösung seines Problems helfen. Weiteren Support bietet Mayo Clinic durch Schulungen, und durch das Angebot an, auf Wunsch des Kunden benutzerdefinierte Module zu entwickeln oder Bildverarbeitungsaufgaben auf einem von ihm bereeitgestellten Datensatz anzuwenden.

[19] Analyze Direct - http://www.analyzedirect.com/

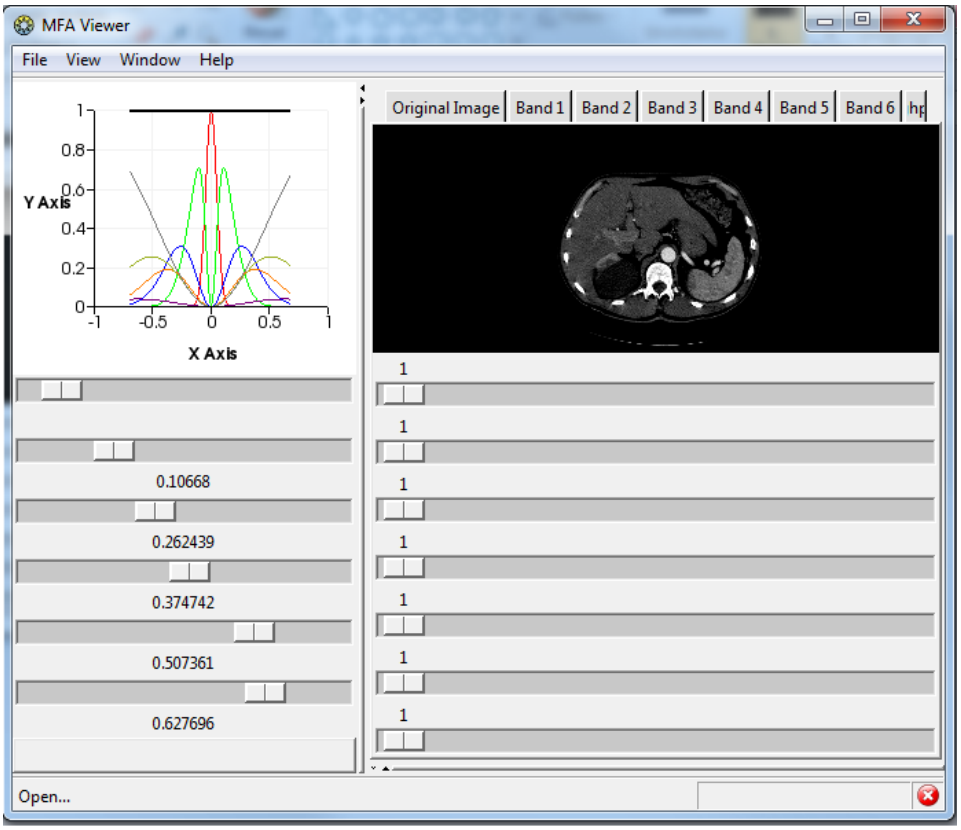

Abbildung 3.32: Das KWWidgets Programm mit AVW Anbindung.

3.7.7 Abschließende Bemerkungen

Analyze ist ein bereits sehr lange im Umlauf befindliches Produkt, mit vielen Versionen und einer langen Entstehungsgeschichte, die bis in die 1970er zurückgeht. Dies ist auch eines der größten Probleme bei Analyze: Das Programm wirkt alt! Es gibt keine zusammenhängende Designguideline, kein wirkliches 'Hauptfenster' auf dem sich alle Arbeitsprozesse abspielen und nachvollziehbar sind. Stattdessen sorgt die Verwendung von vielen kleinen Modulen, die sich alle als seperate Fenster mit eigenen Menüleisten starten für einige Verwirrung. Zum Beispiel kann es vorkommen, dass sich durch die Berechnungen eines Moduls in einem anderen etwas verändert, ohne dass es der Benutzer weiß, weil keine Nachricht von Analyze ausgeschickt wird was passiert ist, und sich auch das entsprechende Modul nicht bemerkbar macht. Analyze ist somit ein Produkt zu dem nur Benutzer Zugang finden, die genau aus der Dokumentation wissen, was welches Modul macht, welche Eingaben es benötigt und was es für Konsequenzen hat, wenn eine Rechenoperation ausgeführt wird. Ein weiterer zu bemängelnder Punkt ist die Nichtlinearität der Laufzeit des MFP Algorithmus bei größer werdenden Datensätzen (Wird in Absch.3.8 behandelt). Da Datensätze in der Medizin, je nach Modaltität, bis zu mehreren tausend Pixel in jede Dimension aufweisen können, schneidet hier die Performance von ITK besser ab als die von AVW. Ein weiterer Minuspunkt ist, dass es keinerlei Einsicht in den Quellcode der Algorithmen von AVW gibt. Dem Benutzer wird hier also nur der Blick auf das Endergebnis gewährt, was sich jedoch in der Black Box tut, bleibt verborgen. Es gibt leider keine Möglichkeit die schon bestehende AVW Bibliothek nach den Wünschen des Benutzers umzumodellieren, da die Bibliotheken vorkompiliert sind und nicht verändert werden können. Positiv wiederum ist, dass Analyze ein professionelles Produkt ist, bei dem der Benutzer Garantie über die richtige Wirkungsweise der Algorithmen hat, und dass es viele Module für Spezialanwendungen beinhaltet. Abschließend wäre zu sagen dass ein lizenzpflichtiges Produkt im Allgemeinen besser sein müsste, als ein vergleichbars Produkt ohne Lizenzkosten, um die entstandenen Kosten zu rechtfertigen. Ob dies für Analyze der Fall ist, hängt vom Anwendungskontext des Benutzers ab.

Größe	Zeit (ms)
128 x128	210,58
256 x 256	449,43
512 x 512	1489,93
1024 x 1024	5801,67
2048x 2048	24043,59

Tabelle 3.2: Laufzeiten des Slicer3 Loadable Moduls

3.8 Laufzeitberechnungen & Bekannte Fehler

Im folgenden Abschnitt soll versucht werden, eine Abschätzung des Rechenzeitbedarfes der MFP Plugins zu geben. Dabei wurde jedes der implementierten Plugins für Slicer3, OsiriX und das mithilfe der AVW Bibliothek erstellte Programm mit 5 float Datensätzen (128x128, 256x256, 512x512, 1024x1024 und 2048x2048) getestet und ihrer Ergebnisse miteinander verglichen. Die abgebildeten Grafiken entsprechen dabei immer den Messwerten, die als Mittel von 100 Iterationen genommen wurden. Dabei wurde ein kompletter Ablauf des jeweiligen Algorithmus für die Zeitmessung verwendet. Dies beginnt beim Laden der Daten bis zum Wiederzurückschreiben des Outputs, wobei alle Bänder miteinbezogen wurden.

3.8.1 Slicer3 Modul

Bei Slicer3 wurden sowohl die Laufzeit des CLP Modul als auch die des loadable Moduls verglichen. Der Vergleich ist insofern interessant, da loadable Module als dll und CLP Module als ausführbare Dateien erstellt werden, und somit komplett anders vom Prozessor intepretiert werden. Eine ausführbare Datei wird als neuer Prozess gestartet, kann alle nötigen Klassen und Bibliotheken beinhalten und somit keine Ressourcen von aussen benötigen, um zu funktionieren. Zusätzlich fordert sie ihren eigenen Speicher an. Eine dll hingegen kann nur durch ein Programm gestartet werden, und läuft dort in einem der Prozesse des Programmes. Den Speicher bekommt die dll Applikation von der Anwendung, die die dll gestartet hat, zugewiesen. Abb. 3.34 zeigt das Laufzeitverhalten des CLP Moduls, Abb.3.33 das des Loadable Moduls, mit den genauen Messungen in Tab. 3.3 bzw. Tab. 3.2. Loadable und CLP Modul, basieren auf der verwendeten ITK Pipeline aus Absch.3.2.1.

Wie man sehen kann ist die Laufzeit des Algorithmus fast linear, das heißt eine Verdopplung der Bildgröße, die einer Verviefachung der Pixelanzahl entspricht, geht einher mit einer ca. ebensogroßen Erhöhung der Laufzeit des Algorithmus. Bei der absoluten Berechnungszeit

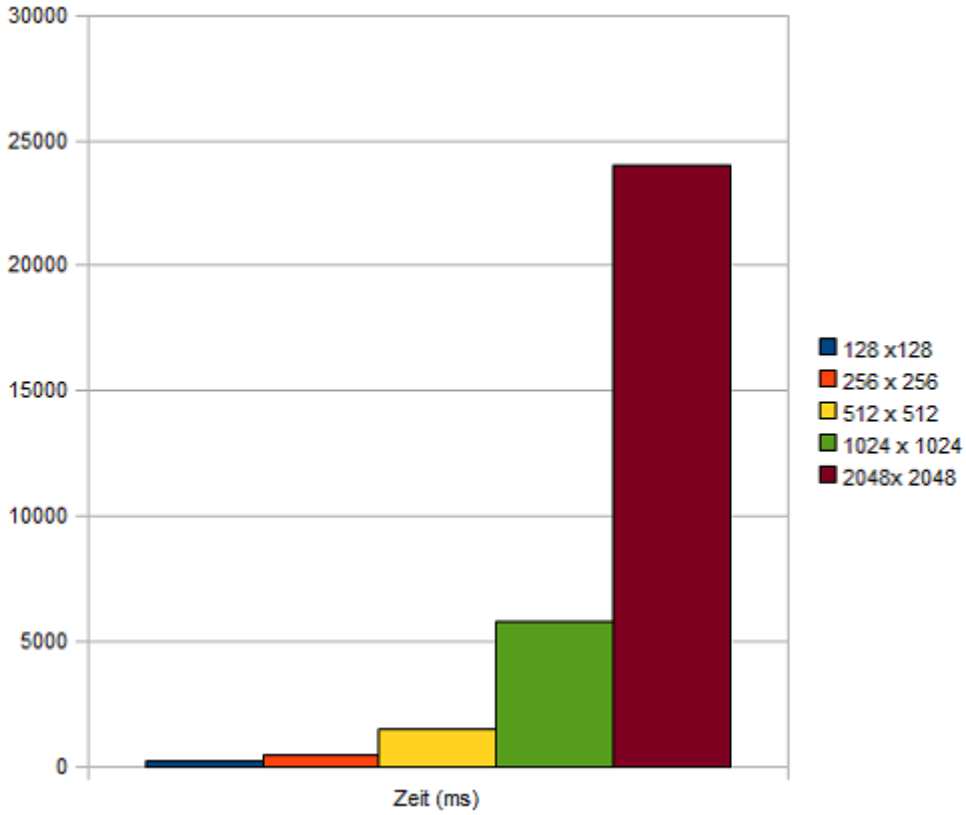

Abbildung 3.33: Laufzeitverhalten des Slicer3 Loadable Moduls.

Größe	Zeit (ms)
128 x128	34,72
256 x 256	102,44
512 x 512	421,94
1024 x 1024	1620,86
2048x 2048	6683,36

Tabelle 3.3: Laufzeiten des Slicer3 CLP Moduls

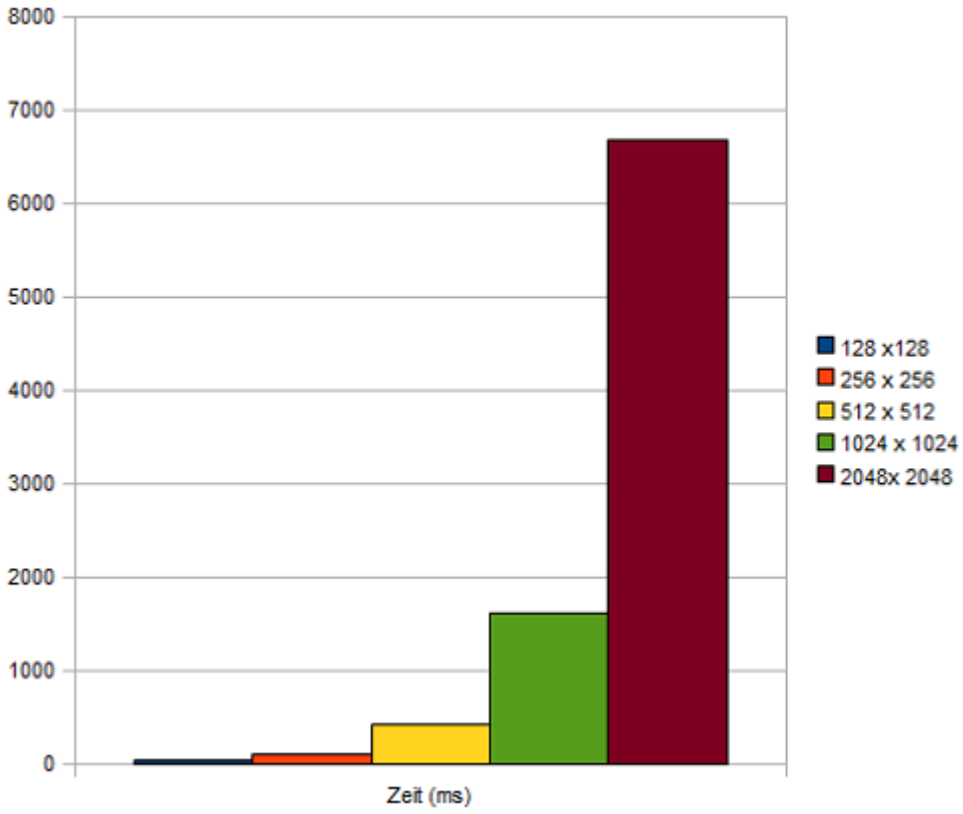

Abbildung 3.34: Laufzeitverhalten des Slicer3 CLP Moduls.

zwischen den zwei Methoden gibt es allerdings gravierende Unterschiede: Der CLP Algorithmus ist fast immer um einen Faktor von 4 schneller als sein dll Gegenstück. Dies entspricht nicht den allgemeinen Erwartungen, nach denen die Ausführung einer dll weniger Zeit in Anspruch nimmt als die einer ausführbaren Datei. Zwar könnte man argumentieren, dass der Ablauf der Algorithmen verschieden ist, und das loadable Modul noch zusätzlich eine Konvertierung zwischen vtkImageData und itkImage vornehmen muss, aber dies kann lediglich eine Verändeurng von ein paar Prozent bewirken. Des Weiteren wurde bei der Ausführung des loadable Moduls ein Speicherproblem entdeckt, das bei dem größten Datensatz (2048 x 2048) zu einem Absturz des Moduls nach ca. 85 Durchläufen führte. Das kommt vermutlich zustande, da sich Slicer3 und loadable Modul gemeinsamen Speicherplatz teilen, und das loadable Modul von Slicer3 eine gewissen Anteil davon zugewiesen bekommt. Eine Erklärung dafür könnte sein, dass alle Dateien als 32 Bit Applikationen vorliegen, und sie daher auch nur eine kleinere Menge an Speicherplatz zur Verfügung gestellt bekommen. Mögliche Problemlösungsstrategien sind in Absch. 3.2.1 näher erläutert. Anonsten kann dies dazu führen dass der benötigte Speicherplatz zu groß wird und die dll einen kritischen Fehler liefert. Das CLP Modul hingegen kann vom System soviel Speicherplatz anfordern wie benötigt, weswegen es hier zu keinen Abstürzen kommt. Wenn man die Filter so umgestaltet dass sie Daten streamen, also nur eine eine kleine Teilmenge des Datensatzes verarbeiten, bzw. den selben Datenspeicher benutzen, sollte man obiges Problem in den Griff bekommen.

3.8.2 OsiriX Plugin

Abb. 3.35 zeigt das Laufzeitverhalten des OsiriX Plugins, mit den genauen Werten in Tab.3.4. Wie man sehen kann sind die Ergebnisse in etwa vergleichbar mit denen des Slicer Loadable Moduls. Unterschiede kommen durch den leicht veränderte Ablauf des Algorithmus, sowie durch andere Hardwareeinstellungen zustande. Die Laufzeitberrechungen wurden auf einem iMac i7 mit 8GB Ram, Snow Leopard 10.5 durchgeführt. Das OsiriX Modul basiert auf der ITK Pipeline aus Absch. 3.2.1. Wie man sehen kann sind hier die Laufzeiten, ähnlich wie bei den Slicer Modulen, auch annähernd linear, oder sogar schneller.

3.8.3 AVW Plugin

Abb. 3.36 gibt einen Überblick über die Laufzeitberechnnug, mit den genauen Werte in Tab. 3.5.

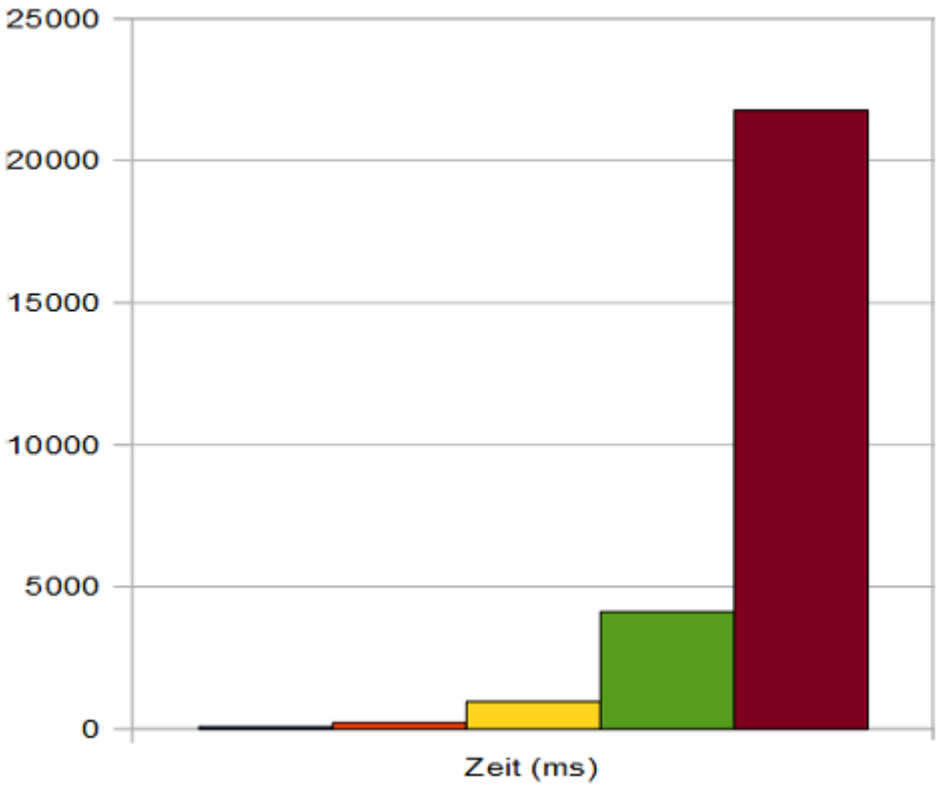

Abbildung 3.35: Diagramm über die Laufzeit des OsiriX Plugins.

Größe	Zeit (ms)
128 x 128	81,32
256 x 256	215,08
512 x 512	971,04
1024 x 1024	4127,78
2048x 2048	21767,87

Tabelle 3.4: Laufzeiten (in ms) des OsiriuX Plugins.

Größe	Zeit (ms)
128 x 128	19,51
256 x 256	89,34
512 x 512	493,19
1024 x 1024	3388,4
2048 x 2048	27497

Tabelle 3.5: Messwerte (in ms) des AVW Plugins

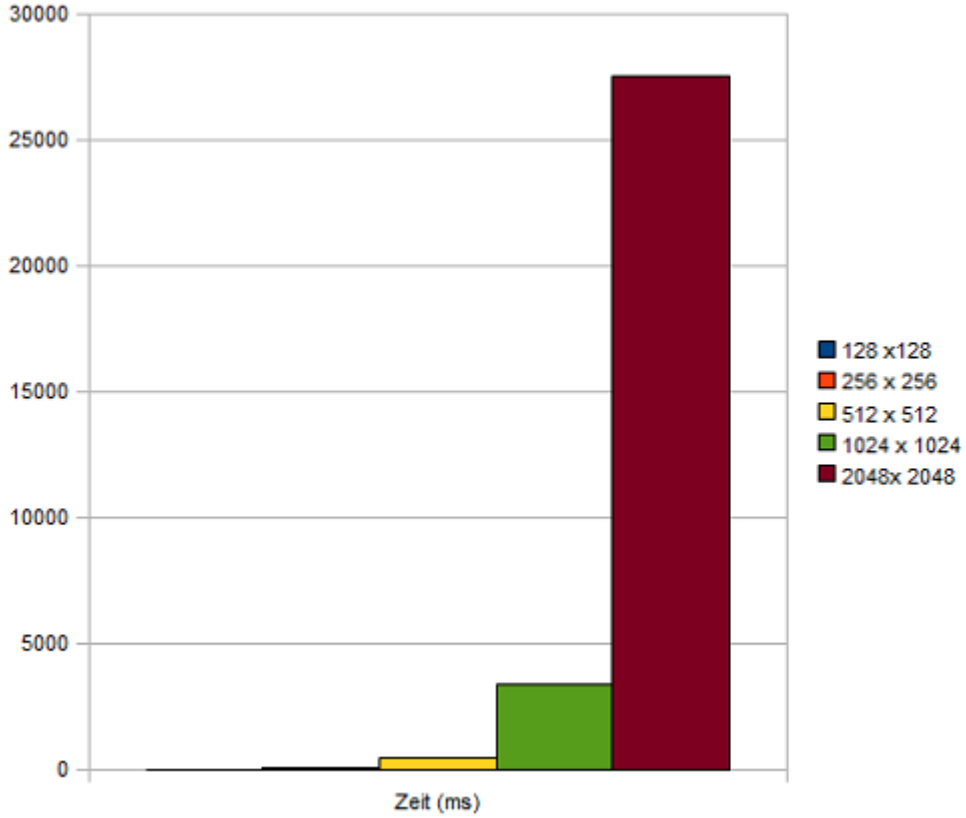

Abbildung 3.36: Laufzeiten für das AVW Plugin

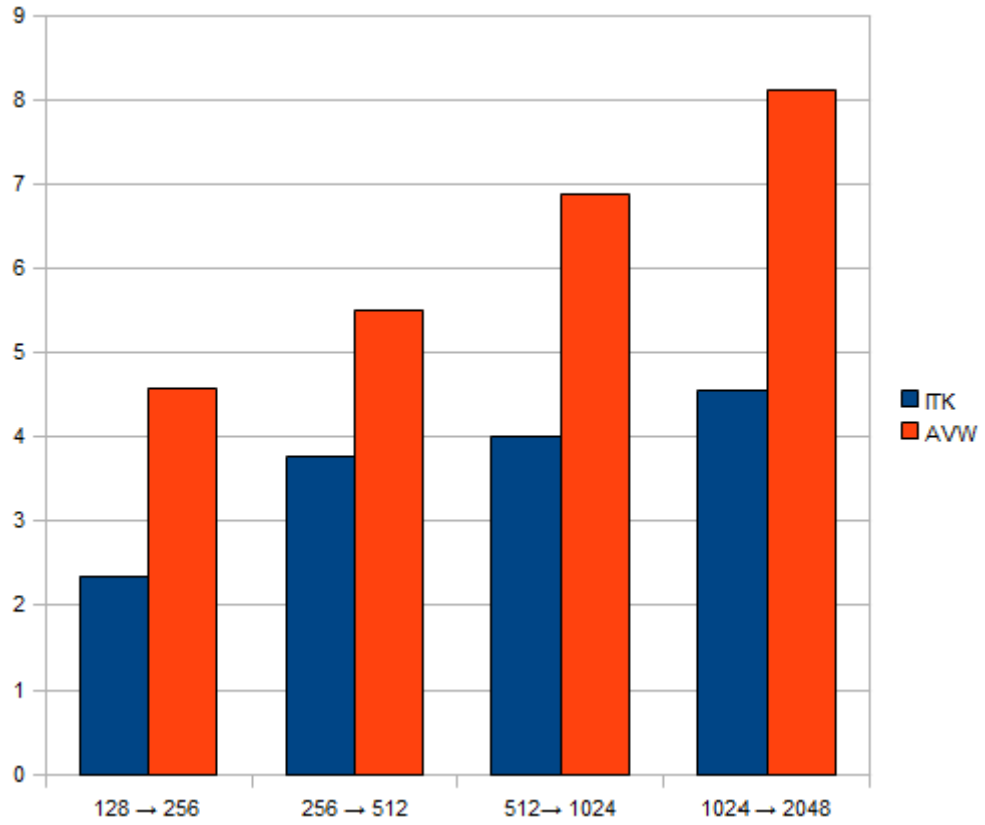

Abbildung 3.37: ITK und AVW Berechnugsfaktoren für die implementierten Plugins.

Die Messwerte zeigen, dass sich das Laufzeitverhalten von AVW in keiner Weise linear verhält. Eine Verdopplung des Datensatzes geht ab einer gewissen Größe nicht mehr mit einer Vervierfachung der Berechnungszeit einher. Dies macht der Sprung von 512 x 512 auf 1024 x 1024, bzw. von 1024 x 1204 auf 2048 x 2048 deutlich. Im ersten Fall entspricht dies einem Faktor von 6, im zweiten einem Falle von 8. Unabhängig von der Gesamtberechungszeit, welche aufgrund verschiedener Hardwarekonfigurationen zustandekommt, kann man hier eindeutig feststellen dass AVW bei der Berechnung größerer Datensätze in schlechteres Laufzeitverhalten für den MFP Algorithmus aufweist. Abb 3.37 gibt die aus den mit ITK und AVW geschriebenen Programmen gewonnenen Laufzeitberechnungen an. Die Grafik gibt nur die Unterschiedsfaktoren an, die entstehen wenn man die Laufzeit, die für einen Datensatz gebraucht wird mit der eines doppelt so großen Datensatzes vergleicht. Wie man sehen kann entsteht für ITK eine annähernd lineare Berechnungszeit, mit allen Faktoren, bis auf den ersten, nahe bei 4, und für AVW ein annähernd linearer Anstieg der Berechnungsfaktoren. Die Faktoren bei den ITK Berechnungen basieren auf den Mittelwerten aller erhaltenenen ITK Berechnungszeiten.

Kapitel 4

Conclusio und Aussichten

Im Verlauf dieser Arbeit wurden 3 medizinische Bildverarbeitungsprogramme/Bibliotheken, miteinander verglichen. Die zu untersuchenden Aspekte betrafen die bereits implementierten Algorithmen, die Benutzerfreundlichkeit, die Dokumentation, sowie die Erweiterbarkeit dieser Programme durch ein selbstgeschriebenes Plugin. Slicer3D ist eine frei verfügbare Software, für Mac, Windows, Linux, und Solaris Systeme. Die speziellen Stärken liegen in Registration, Segmentation und Visualisierung und Bildbearbeitung einer Vielzahl von Bildtdateiypen. Slicer3 kann wahlweise in 32 Bit oder 64 Bit kompiliert werden; ein großer Pluspunkt, da ein 64 Bit System die immer größer werdende Datenflut und aufwändigere Algorithmen mit mehr benötigtem Speicherplatz besser verarbeiten kann. Konzeptuell unterscheidet sich Slicer3 von den anderen hier untersuchten Programmen, da es eine eigene XML ähnliche Scriptsprache (MRML) für die Kommunikation zwischen Plugins benutzt. Plugins können somit untereinander kommunizieren und sich Daten zuschieben, was benutzt werden kann, um eine regelrechte Verarbeitungspipeline mehrere Plugins auszuführen. MRML fließt auch in den programmiertechnischen Standpunkt von Slicer3 ein. Für einfache Module erzeugt der Benutzer mit MRML eine GUI, die in Slicer3 integriert werden kann. Das eigentliche Programm ist eine simple ausführbare Datei. Damit lassen sich Plugins relativ schnell und einfach in Slicer3 implementieren. Für komplexere Module muss sich der Programmierer näher mit der MRML Syntax und dem Execution Model auseinandersetzen. Seine Funktionalität bezieht Slicer3 aus einer Anzahl kleinerer Frameworks. Diese Frameworks liegen vollständig in der Installation vor. Frameworks können z.B: verändert werden, da sie bei jedem Systemstart von außen in Slicer3 inkludiert werden. Slicer3 ist nicht zur Befundung zugelassen, und kann somit ausschließlich in der klinischen Forschung eingesetzt werden. Nichtsdestotrotz erfreut sich Slicer einer großen und aktiven Community, die für Fragen aller Art offen ist, und

ständig um Verbesserung und Erweiterung von Slicer3 bemüht ist. Auch die Dokumentation von Slicer3 ist sehr gut, speziell was den Einsatz der Module in Zusammenhang mit der MRML Syntax anbelangt. Leider verfügt Slicer3 nicht über die Möglichkeit einer Pacs Anbindung, was den Datenimport in Slicer etwas umständlich macht. Die zweite untersuchte Bildverarbeitunsgsoftware, OsiriX, ist ausschließlich für Mac Systeme verfügbar. Von OsiriX gibt es mehrere Versionen: Die Normalversion ist 32 Bit stark, und steht uneingeschränkt und gratis zur Verfügung. Die 64 Bit Erweiterung ist kostenpflichtig, bietet allerdings auch alle damit einhergehenden Schnelligkeitsvorteile. Ebenfalls kostenpflichtig ist die Osirix MD Version, welche in den USA eine FDA und in Europa eine CE-1 Kennzeichnug besitzt und für den Einsatz in der Befundung und Diagnose zugelassen ist. Osirix HD ist für iPad/iPhone frei verfügbar. Zuguterletzt gibt es noch Osirix Lite, welches eine auf CD Rom verfügbare und dementsprechend abggespeckte Version ist. OsiriX ist ebenso wie Slicer3 anwendbar für Registration, Segmentation, Bildbearbeitung und visueller Darstellung von vielen verschiedenen Bilddatenformaten. Anders jedoch als bei Slicer3 liegen die von OsiriX verwendeten Frameworks bereits vorkompiliert vor, und können nur unter Einschränkungen verändert werden. Ein entscheidener Nachteil, da die Frameworks teilweise schneller aktualisiert werden, als die statischen in OsiriX enthaltenen, und OsiriX somit immer ein bißchen veraltet sein kann. Zwar gibt es findige Entwickler, die versucht haben dieses Problem zu umgehen, und OsiriX eine eigene Version eines bestimmten Frameworks verwenden zu lassen, allerdings ist dies nur sehr vereinzelt möglich.OsiriX besitzt zusätzlich zu seinen Qualiäten als Bildverarbeitunsgsoftware noch eine PACS Anbindung, d.h. einen entsprechend konfigurierten Server vorausgesetzt, dient OsiriX dazu Bilddaten zu verschicken, zu empfangen und zu archivieren. Vom programmiertechnischen Standpunkt ist OsiriX sehr leicht zu bedienen. Da OsiriX auf Mac Systemen läuft, benutzt es für die Programmierung Apples eigene Entwicklungsumgebung, XCode. Zusätzlich gibt es von den OsiriX Entwicklern einen Plugin Generator, der ein entsprechend vorkonfiguriertes, leeres XCode Projekt erzeugt. Die GUI selbst wird über den XCode Designer mit dem 'Drag & Drop' Prinzip zusammengesetzt. OsiriX besitzt ein kleines eigenes Framework, das die wichtigsten Programmierbausteine enthält (Bilder, Anzeigebildschirme), was dem Entwickler den Einstieg verinfacht. Leider ist es allerdings in OsiriX für Module nicht möglich miteinander zu kommunizieren. Eine Verarbeitunspipeline mehrere Module ist somit nicht möglich. Nachteile von OsiriX sind die Community und der Support von OsiriX: Die Newsgroups (französisch / englisch) sind nicht sehr aktiv, Fragen werden nicht beantwortet, und oftmals wird man mit seinen Problemen alleingelassen. Allerdings ist die Dokumentation von OsiriX ausreichend gut, sodass man mit ein bißchen Aufwand durchaus zur Lösung seiner Probleme kommt. Ein weiterer Nachteil ist die teils erschwerte Möglich-

keit, beim Programmieren von OsiriX Plugins, die Frameworks von OsiriX einzubinden. Die dritte Bildverarbeitunsgsoftware, Analyze, ist lizenzpflichtig und für Windows, Linux, Mac und Solaris (32 / 64 Bit) gleichermassen verfügbar. Analyze besitzt Funktionalitäten in Segmentation, Registration, Bildbearbeitung und Visueller Darstellung, und besitzt auch eine Pacs Anbindung. Analyze besitzt keine zusammenhängene Benutzeroberfläche, stattdessen nutzt es exzessiv das Konzept von Modulen, kleinen seperaten, funktionellen Bauelementen. Dies endet in einem Übersichtschaos: Jede Funktion öffnet ein seperates Fenster. Bei mehreren offenen Fenstern ist es nicht einfach das richtige wiederzufinden, wenn dieses überlappt wird. Jedes dieser Fenster besitzt zusätzlich eine eigene Menüleiste, mit teils doppelter Verwendung der gleichen Bildsymbole, aber mit unterschiedlicher Funktion. Analyze bezieht seine Funktionalität größtenteils aus der hauseigenen AVW Bibliothek, die ebenfalls lizenzpflichtig ist. Die Dokumentation ist leider teils unvollständig, bzw. lassen sich gewisse Probleme auf eine Art lösen, auf die nicht in der Dokumentation hingewiesen wird. Dies ist umso bedenklicher, da die AVW Library vorkompiliert ist, und dem Programmierer keinen genauen Einblick in die Algorithmen oder Verarbeitunsgprozesse erlaubt. Der Support für Analyze / AVW wird mit einer neuen Lizenz für eine gewisse Zeitspanne erworben, danach kostet jede Anfrage beim Support Team extra. Generell benötigt man für Analyze überall eine Lizenz: Eine Lizenz für Analyze, eine Lizenz für AVW, falls man die Bibliothek getrennt verwenden will, und Lizenzen für bestimmte Module. Ein eigenes Modul für Analyze zu schreiben ist somit nur möglich, wenn man die Lizenz für ein Erweiterungsmodul erworben hat.

Zusammenfassend wäre also zu sagen:

Slicer3 profitiert von einer sehr großen Community, von bewährten Frameworks, und von einem durchdachten Programmkonzept (MRML), das es unter anderem Modulen erlaubt miteinander eine Verarbeitunspipeline zu bilden, sowie von der Verfügbarkeit als 32 / 64 Bit Version. Negativ zu beurteilen ist die Konversion zwischen VTK Daten und ITK Daten, da eines als Darstellungsdatentyp und das andere als Bearbeitunsgdatentyp dient. Weiters ist das MRML Konzept zwar ein gut durchdachtes Designkonzept, wohl allerdings für die meisten Anwendungn zu komplex als dass es richtig genutzt werden könnte. Slicer3 fehlt auch die Anbindung an ein Pacs System, und hat hier einen Nachteil OsiriX und Analyze gegenüber.

OsiriX profitiert von einem guten Programmkonzept, und übersichtlicher Gestaltung der Programmelemente, sowie von bewährten Frameworks und einer guten Benutzerführung. Erweiterungen über Plugins sind, dank der guten Dokumentation und der XCode Program-

mierumgebung relativ leicht umzusetzen. Negativ fällt auf, dass nur die 32 Bit Version gratis verfügbar ist, dass Module nicht miteinander kommunizieren können, und dass mitgelieferte Frameworks teilweise veraltet sind (jedoch mittels Anleitung teilweise händisch upgedatet werden können), sowie dass die Integration dieser Frameworks bei der Entwicklung von neuen Plugins nicht ganz einfach ist, und es nur schlechte Hilfestellungen aus der Dokumentation hierzu gibt.

Analyze hat gegenüber Slicer3 und OsiriX keine wesentlichen Vorteile: Es ist kostenpflichtig, das Design des Programms ist für den Benutzer verwirrend, und es bietet sich kein Einblick in die Black Box der Verarbeitungsalgorithmen von AVW. Erweiterungen am eigentlichen Programm können nur mittels kostenpflichter Module vorgenommen werden. Die einzigen Pluspunkte, die Analyze vor Slicer3 und OsiriX hervorheben, sind der wohl professionelste Support, die Garantie auf Richtigkeit der Algorithmen, sowie Module für spezielle Anwendungsfälle.

Aus persönlicher Sicht ist die Entscheidung, welches der 3 in dieser Arbeit vorgestellten Programme das Beste ist, nur eine Entscheidung zwischen OsiriX und Slicer3. Analyze hat aufgrund oben genannter Eigenschaften einen entscheidenden Nachteil gegenüber den beiden, und scheidet somit im direkten Vergleich aus. Slicer3 ist ein ambitioniertes Projekt mit einer großen Entwicklercommunity. Ein goßer Vorteil, da dies, nach persönlicher Meinung, eine gute Basis für ständige Weiterentwicklungen hinsichtlich Funktionalität und Performance darstellt. Die Community von OsiriX ist leider sehr verschwiegen, Antworten kommen nur zögerlich, und sind auch nicht unbedingt hilfreich. Im direkten Vergleich schneidet hier OsiriX deutlich schlechter ab. Vom programmiertechnischen Standpunkt unterscheiden sich OsiriX und Analyze grundlegend. Da OsiriX nur auf Mac Systemen läuft, kann ganz einfach Apple XCode für die Implementation neuer Plugins verwendet werden. Die Entwickler von OsiriX haben Bedacht darauf gelegt, dass das Programmieren neuer Plugins möglichst einfach gemacht wird, indem sie einen 'Plugin Generator' zur Verfügung stellen, der alle notwendigen Einstellungen vornimmt um ein Plugin zu erzeugen. Slicer hingegen läuft auf Solaris/ Unix/windows und Mac Systemen, und vergrößert somit schoneinmal die Menge der Leute, die potentielle Slicer Benutzer sein könnten. Der Programmierer muss sich bei Slicer3 erst in das Konzept des Slicer Execution Models und der MRML Scriptsprache einlesen. Zusätzlich gibt es hier die Auswahl zwischen mehreren Sorten von Modulen, was ansich eine gute Idee ist, aber zum schnellen Starten in Slicer3 unbrauchbar ist. Also ergibt sich hinsichtlich Verfügbarkeit für Slicer3 ein PLuspunkt und hinsichtlich Einfachkeit beim Programmieren einer für OsiriX. Ein weiterer Nachteil von Slicer3 ist, dass selbstgeschriebene Plugins nur auf Sli-

cer3 Versionen aufgerufen werden können, die mit demselben Kompiler erstellt wurden, wie das Plugin selbst. Die Verwendung setzt also vorraus, daß diese Slicer3 Versionen ebenfalls selbst kompiliert wurden. Eine Integration in die vorkompilierte Slicer3 Version ist nur möglich, wenn das Plugin beim Slicer Development Team eingereicht und die Verwendung bewilligt wurde. (Dazu muss sich das Plugin streng an das MRML Konzept halten). In OsiriX hingegen gibt es keine Probleme bei der Importation eines selbstgeschriebenen Plugins in eine andere OsiriX Version, da auch nur mit einer Entwicklungsumgebung, XCode gearbeitet wird. Wurde ein Plugin einmal kompiliert, kann es in jeder OsiriX Version installiert und aufgerufen werden. Ein Nachteil bei der Programmierung von Plugins in OsiriX ist allerdings die schlechte Möglichkeit Frameworks zu verwenden, bzw. auszutauschen, da diese bereits vorkompiliert, und oftmals auch in ihrem Umfang reduziert sind. Seitens der Entwickler gibts es hierzu auch kaum Hilfestellungen, wie z.B Frameworks wie ITK in OsiriX Plugins einzubinden sind. Hier wiederum hat Sllicer3 einen großen Vorteil, weil alle Frameworks seperat vorliegen und zur Laufzeit erst eingebunden werden. Somit ergibt sich nicht nur die Möglichkeit auf die gesamte Funktionalität eines Framworks zuzugreifen, sondern auch, diese jederzeit gegen z.B: eine neuere Version auszutauschen. Vorteilhaft ist für Slicer ausserdem die Tatsache, dass Module mithilfe des MRML Protokolls miteinander kommunizieren, und so eine Verarbeitungspipeline bilden können. Für OsiriX ist dies nicht möglich; jedes Plugin kann andere Plugins nicht ansprechen und existiert nur für sich. Einen großen Nachteil in Punkto Funktioanlität besitzt Slicer gegenüber OsiriX jedoch in der Tatsache, dass Slicer3 ein Pacs System fehlt, und dadurch der Datenimport, bzw. die Kommunikation mit anderen Slicer3 Versionen erschwert wird. Hinsichtlich der Geschwindigkeit für Verarbeitungsprozesse wurden im Verlauf dieser Arbeit zwar nur die 32 Bit Versionen getestet, allerdings gibt es für Slicer3 die Möglichkeit auch auf 64 Bit zu kompilieren, wohingegen bei OsiriX die 64 Bit Version kostenpflichtigt ist. Abschließend wäre zu sagen, dass es nach persönlicher Meinung nur davon abhängt auf welche der obigen Punkte mehr Bedacht gelegt wird, um zu entscheiden, welche der zwei Softwareprodukte für weitere Projekte eingesetzt werden soll. Beide Produkte haben Stärken und Schwächen, und das eine hat Probleme, wo das andere punkten kann, und umgekehrt.

Kapitel 5

Literaturverzeichnis

[1] Wolfgang Bikfellner. *Applied Medical Image Processing: A Basic Course*. CRC Press, 2010.

[2] Mayo Clinic. Analyze 10.0 tutorial manual cd. 2009.

[3] W. Hillen. Praktikum 5 digitale bildverarbeitung: Bildfilter (teil ii), 2005.

[4] Copyright http://www.vegan hund.de, 2011.

[5] Bernd Jähne. *Digitale Bildverarbeitung*. Springer Verlag Berlin Heidelberg New York, 1991. ISBN 3-540-53768-6.

[6] Copyright Hans Kellner, 2005.

[7] C.M Schaefer-Prokop M. Prokop. Digital image processing. *Europ. Radiol. 7*, 1997.

[8] Jim Miller. Slicer3 gui generation: A tour of the execution model xml, 2011.

[9] R. E. Woods R. C. Gonzalez. *Digital image processing*. Prentice Hall, 2008.

[10] Robb RA. The biomedical imaging resource at mayo clinic. guest editorial. *IEEE Transactions on Medical Imaging*, 2001.

[11] Karwoski RA Larson AG Workman EL Stacy MC Robb RA, Hanson DP. Analyze: A comprehensive, operator-interactive software package for multidimensional medical image display and analysis. *Coputerized Medical Imaging and Graphics 13*, 1989.

[12] Slicer Development Team. Building a slicer 3 module gui, 2007.

[13] The OsiriX Development Team. Creating a plugin: Keynote presentation, 2008.

[14] Schoeters E Vuylsteke P. Multiscale image contrast amplification. *SPIE Medical Image Proc.*, 1994.

[15] Wikipedia. Nachbarschaft (bildverarbeitung), 2006.

[16] Wikipedia. Point spread function, 2006.

[17] Mark James Burge Wilhelm Burger. *Digitale Bildverarbeitung: Eine Einführung mit Java und ImageJ*. Springer, 2006.