Technische Hochschule Augsburg

Fakultät für Informatik

Bachelorarbeit

zur Erlangung des Grades
Bachelor of Science in Informatik

Automatische Segmentierung von Micro-CT Bildern zur Untersuchung zahnmedizinischen Strukturen

erstellt von Lukas Konietzka

Lukas Konietzka

Sebastian-Kneipp-Gasse 6A
86152 Augsburg
T +49 172-2728-376
lukas.konietzka@tha.de

Matrikelnummer:
2122553

Technische Hochschule Augsburg

An der Hochschule 1
D-86161 Augsburg
T +49 821 5586-0
F +49 821 5586-3222
www.tha.de
info@tha.de

Erstprüfer	Prof. Dr. Peter Rösch
Zweitprüfer	Prof. Dr. Gundolf Kiefer
Eingereicht am	November 14, 2024
Verteidigung am	März 20, 2025
Geheimhaltungsvereinbarung	Nein

Kurzfassung

bla

Inhaltsverzeichnis

1. Einleitung

Die Computertomografie (CT) hat die Medizintechnik revolutioniert und ist bis heute eines der wichtigsten Methoden für die Bildanalyse. Sie ist eine der führenden Erweiterungen der klassischen Röntgentechnik. Für die Entwicklung dieser Technologie wurden Godfrey Newbold Hounsfield und Allan McLeod Cormack im Jahre 1979 mit dem Nobelpreis für Medizin ausgezeichnet (Handels 2000, Seite12).

Die Computertomografie wird in den verschiedensten Bereichen und im wahrsten Sinne des Wortes von Kopf bis Fuß eingesetzt. So kommt es, dass auch im Dentalbereich CT Aufnahmen von größter Wichtigkeit sind. Abbildung 1.1 zeigt eine solche CT-Aufnahem. Eine konkrete Anwendung in diesem Kontext ist die Zahnkaries Forschung der Poliklinik für Zahnerhaltung und Parodontologie des LMU- Klinikums München.

Abbildung 1.1.: *CT-Aufnahme eines Zahns Quelle (Poliklinik 2024)*

Die vorliegende Arbeit soll genau diese Forschung unterstützen. In welchem Umfang und zu welchem Grund ist in den folgenden Abschnitten beschrieben.

1.1. Ziel der Arbeit

Diese Arbeit beschreibt eine Technik, mit der dreidimensionale Micro-CT Bilder zur Untersuchung zahnmedizinischen Strukturen automatisch mittels der Software *3D Slicer* segmentiert und analysiert werden können. Die algorithmische Formulierung einer konkreten Segmentierung ist bereits vorhanden und prototypisch implementiert. Dieser Algorithmus muss jedoch umständlich über ein Kommando im Terminal ausgeführt werden, was die Benutzerfreundlichkeit deutlich beeinträchtigt. Ziel dieser Arbeit ist es nun das bereits implementiert Verfahren automatisiert über ein interaktives User

Interface (UI) zur Verfügung zu stellen. Dabei soll auf etablierte und vertraute Lösungen zurückgegriffen werden.

Es stellt sich nun die Frage, zu welchem Zweck eine automatische und interaktive Segmentierung überhaupt notwendig ist. Für die Zahnklinik an der LMU in München gibt es hierfür viele Gründe. Über den wichtigsten gibt das nächste Kapitel Aufschluss.

1.2. Relevanz der Arbeit

Der wohl relevanteste Punkt dieser Arbeit ist, dass Ärzte reine Anwender und keine Entwickler von Software sind. Darüber hinaus verfolgt die Parodontologie der LMU in München einen sehr interessanten Forschungsansatz, welche eine Segmentbetrachtung der CTs unumgänglich macht.

Über viele Jahre hinweg wurden in der Zahnklinik sehr viel Bilddaten von Zähnen gesammelt, die aufgrund von Zahnkaries entfernt wurden. Hierbei wurden Aufnahmen der unterschiedlichsten Arten gemacht. Darunter fallen zum Beispiel einfache Bilddateien, Infrarotbilder und die für diese Arbeit so relevanten dreidimensionalen Micro-CT Aufnahmen. Dieser große Schatz an Bildmaterial soll verwendet werden, um in ferner Zukunft ein neuronales Netzwerk zu trainieren, welches statistische Aussagen über das Verhalten von Karies treffen kann. Jedoch gibt es hier ein Problem, bei dem das Ergebnis dieser Arbeit Unterstützen kann.

Karies auf CT-Bildern zu lokalisieren ist nicht trivial. Er ist ohne weitere Bearbeitung des Bildes nur sehr schwer auf eine Stelle einzugrenzen. So kommt es vor, dass drei verschiedene Ärzte auf dem selben Micro-CT Bild drei unterschiedliche Stellen mit Karies identifizieren. Eine Segmentierung des dreidimensionalen CTs kann hier Wunder wirken lassen. Durch die Aufteilung des Micro-CTs in seine zwei Zahnhauptsubstanzen, kann in das innere der Zähne geblickt werden, was die Lokalisierung kariöser Stellen deutlich vereinfacht.

Mit dieser klaren und einduetigen identifizierung von Karies, sind die Ergebnisse, die ein neuronales Netzer generieren würde viel genauer und brauchbarer. Konkret wird mit einer automatischen Segmentierung ein *Ground Trueth* gewonnen, der eine eindeutige Basiswahrheit liefert.

Hierbe sei gesagt das diese Anwendung nur ein von vielen Möglichkeiten ist. Konkrete Daten über die Ausbreitung einer Krankheit im Menschlichen Körper zu besitzten kann in den verschiedensten Fällen und Institutionen von größtem Nutzen sein. So zeigen es auch Crespigny u. a. (2008) in ihrem Paper. Dieses Argument stand dehmnach auch für diese

Arbeit bereits zu Beginn im Mittelpunkt und bildet somit den Fokus der Untersuchung, welcher im Nachfolgenden Kapitel näher beleuchtet werden soll.

1.3. Fokus der Arbeit

Für eine automatische Segmentierung von Micro-CT Bildern gibt es einige Softwarelösungen am Markt, die alle eine gut optionen sind. Dieser Arbeit setzt den Fokus auf die Open Sorce Plattform *3D Slicer*, da diese ohnehin bereits eine breite Anwendung in der Zahnklinik in München findet. Durch die Modul und Plugin Infrastruktur dieser Plattform kann die Softwar auch anderen Institutionen bereitgestellt werden. Hierzu muss diese einfach als *3D Slicer Extetion* bereitgestellt werden. *3DSlicer* bietet einen Extension Manager, der ähnlich wie ein App Store betrachtet werden kann. So bleibt die vorerst konkret entwickelte Software nicht nur einer Einrichtung vorbehalten.

Diese Arbeit setzt so den Fokus auf die Extention von *3D Slicer* um so eine automatische und interkative Schnittstelle zu gewähren. Die Optimierung des bereits bestehenden Verfahrens wird nicht thematisiert.

Mit dieser Umfang, der Motiavtion und dem gesetzten Fokus, ergibt sich für dies Arbeit eine konkrete Struktur die nun kurz erläutert werden soll.

1.4. Aufbau der Arbeit

Die Arbeit ist in sieben Kapitel unterteilt. Nach der Einführung in Kapitel 1, in der die Relevanz und der Fokus beschrieben werden, werden in Kapitel 2 die theoretischen und technischen Grundlagen behandelt, welche zum Verstehen der Ergebnisse essenziell sind. Als Ergebnis der theoretischen Grundlagen bildet das Kapitel 3 eine konkrete Forschungsfrage. Während sich Kapitel 4 darum kümmert mit welchen Methodiken und Lösungsansätzen an die Forschungsfrage herrangegangen wird, erläutert das Kapitel 5 was die konkreten Ergebnisse der Arbeit sind. In Kapitel 6 erfolgt eine kritische Diskussio der Resultate einschließich möglicher Limitationen. Das Abschließende Kapitel7 fasst die wichtigsten Erkenntnisse zusammen und gibt einen Ausblick auf zukünftige Forschungsfragen.

Die theoretischen Grundlagen die wie beschrieben nach der Einleitung folgen, sind zentral für das Versetehen der Fragestellung und der methodischen Ausarbeitung.

2. Theoretische Grundlagen

Dieses Kapitel führt in die theoretischen Grundlagen ein, die in dieser Arbeit benötigt werden. Den ersten Teil bilden die domänenspezifischen Grundlagen 2.1, welche genauer darauf eingehen, welchen Inhalt die zu bearbeitenden Bilder bieten und wie dieser zu verstehen ist. Abschnitt 2.2 geht anschließend genauer auf die Bildgebung ein, die eine wichtige Rolle spielen. Der Abschnitt 2.4 geht auf die Arbeit von Hoffmann (2020) ein und legte damit den Grundstein dieser Arbeit. Die Abschnitt 2.5 führt in Softwareentwicklungsthemen ein, die zum Erstellen einer *3D Slicer Extention* wichtig sind.

2.1. Domänenspezifisch

Wie bereits aus dem Kapitel 1 Einleitung klar wurde, handelt es sich bei den Micro-CT Bilder um Zahnbilder, die aufgrund von Zahnkaries entfernt wurden. Um zu verstehen, wie eine CT-Aufnahme eines Zahns aufgeteilt werden soll, ist es hilfreich zu verstehen, wie ein Zahn aufgebaut ist.

Die Abbildung 2.1 zeigt den groben Aufbau eines Zahnes nach K. M. Lehmann u. a. (2012, Seite 17). Zu sehen ist, dass das Denit oder auch Zahnbein genannt, den Großteil eines Zahnes einnimmt. Im Bereich der Zahnkrone wird das Dentin von Zahnschmelz überzogen. Der Zahnschmelz ragt in die Mundhöhle und ist nach K. M. Lehmann u. a. (2012, Seite 41) das härteste Material im menschlichen Körper. In der Mitte des Zahnes befindet sich Weichgewebe, welches als Pulpa bezeichnet wird vgl. (K. M. Lehmann u. a. 2012, Seite).

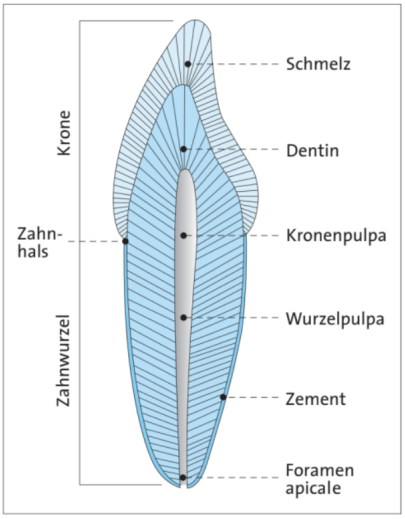

Abbildung 2.1.: *Aufbau eines Zahnes nach K. M. Lehmann u. a. (2012)*

Für die Bearbeitung von Micro-CT Aufnahmen sind die Bereich Schmelz Dentin und Pulpa von besonderer Bedeutung. Betrachtet man eine CT wie es zu Beginn in der Abbildung 1.1 gezeigt wurde, so bilden diese 3 Gewebearten die unterschiedlichen Grauwerte in einem CT-Bild.
Die Pulpa unterscheidet sich hierbei nur wenig vom Hintergrund, da sie als einzige der drei Hauptteile eines Zahnes ein Weichgewebe ist und bei einer Röntgenaufnahme nicht absorbiert. Dieser Teil ist für demanch für die spätere Verarbeitung weniger interessant. Geht man weiter von innen nach außen, so ist der nächste Zahnteil auf einem CT das Zahnbein.
Das Dentin ist laut K. M. Lehmann u. a. (2012, Seite 41) eine Hartsubstanz, die dem Kieferknochen sehr nah steht. So kommt es, dass dieser Teil schon deutlich besser auf einem CT zu erkennen ist. Den äußersten Teil in der Mundhöhle bildet das Zahnschmelz.
Der Schmelz ist wie bereits erwähnt, der härteste Teil im menschlichen Körper und aus diesem Grund auch am hellsten auf dem CT zu erkennen. Die folgende Abbildung 2.2 sollen durch Gegenüberstellung den Zusammenhang zwischen einem CT-Bild und einer Zahnzeichnung verdeutlichen.

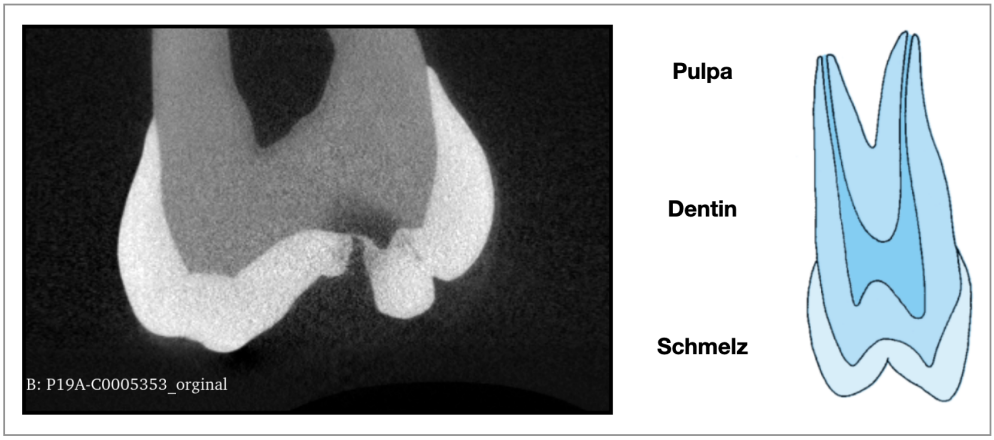

Abbildung 2.2.: *Darstellung von Pulpa, Dentin und Schmelz auf einer CT-Aufnahme (link) und einer Zeichnung (rechts) nach K. M. Lehmann u. a. (2012, Seite 29).*

Mit diesem Domänenwissen kann ein Schritt weiter gegangen werden, sodass der Fokus nun auf die CT-Bilder gesetzt wird. Das Kapitel 2.2 Bildgebung führt die Technologie der Computertomografie tiefer ein. Darüber hinaus werden die verschiedenen Formate und statistische Modelle der CT-Aufnahmen vorgestellt.

2.2. Bildgebung

Es gibt die unterschiedlichsten Arten zur Erzeugung dreidimensonaler Bilddaten. Dieser Abschnitt erläutert federführend die Technologie der Micro-CT Aufnahmen und deren Erstellung. Diese sind für eine medizinischen Einsatz besonders interessant. Des Weiteren erfolgt eine Einführung in die Speicherung und Komprimierung von CT-Aufnahmen. Dies sorgt dafür, dass die digitalen Bilddaten deutlich handlicher werden.

2.2.1. Computertomografie

Die Erfindung der Computertomografie (CT) war ein Quantensprung in der Geschichte der Medizin. Sie ist aus heutigen Diagnosen nicht mehr wegzudenken. Ein Micro-CT Bild ist laut Baird und Taylor (2017, Abstract) ein Menge hochauflösender Bilder, die wie ein Stapel zusammengelegt werden. Der Aspekt Micro deutet dabei darauf hin, dass es eine miniaturisierte Ausführung eines üblichen Kegelstrahl-CTs ist so Buzug (2011, Seite 340). Eine andere Definition erläutert T. Lehmann u. a. (2013). Er beschreibt die Computertomografie als Projektionen einzelner Ebenen im Untersuchungsobjekt. Die Technologie, mit der diese Bilderstapel aufgenommen werden, ist unter der Röntgentechnik oder auch X-Ray bekannt. Die Röntgenstrahlung ist eine Form der elektromagnetischen Strahlung, ähnlich wie das sichtbare Licht so das National Institute of Biomedical Imaging and Bioengineering (NIBIB) (2024). Anders als das Licht haben die Röntgenstrahlen eine viel höhere Energie. Das führt dazu, dass man mit dieser elektromagnetischen Strahlung viele Objekte durchdringen kann. So auch Gewebeteile eines Zahnes vgl. (National Institute of Biomedical Imaging and Bioengineering (NIBIB) 2024). Die Abbildung 2.3 zeigt diese elektromagnetische Spektrum.

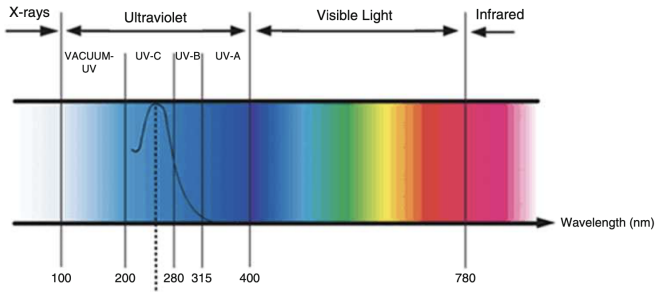

Abbildung 2.3.: *Einordnung der Röntgenstrahlung (X-Ray) nach dem Zwinkels (2015)*

Durchdringt ein solcher Röntgenstrahl ein Untersuchungsobjekt, werden die Details aufgrund der Wechselwirkung mit Materie auf einer CT-Probe sichtbar. Die bekannteste Wechselwirkung ist die Absorption. Mit der Steigerung der Atomzahl in einem Material

nimmt auch die Absorption eines Materials zu, sodass es leicht ist verschiedenen Materialien in einer CT-Aufnahme zu unterscheiden (vgl. National Institute of Biomedical Imaging and Bioengineering (NIBIB) 2024).

Für eine Micro-CT Aufnahme bedarf es spezieller Technik. Es gibt unterschiedliche Firmen, welche die unterschiedlichsten Modelle anbieten. Im Falle der Zahnklinik an der LMU in München handelt es sich um ein Micro-CT 40 der Firma SCANCO Medical AG (2024). Dieses Gerät erstellt Aufnahmen mittels Röntgenstrahlung und generiert mithilfe der Computertomografie ein dreidimensionales Bild, welches im Format `.ISQ` abgelegt wird. Wie das nächste Kapitel beschreiben wird, ist der Speicherumfang den solch ein Bild benötigt, deutlich zu groß. Es bedarf einer Technik, mit der die Aufnahmen auf eine handhabbare Größe schrumpfen.

2.2.2. Datenformate

Die rohen Datensätze, welche direkt aus dem Micro-CT Gerät kommen, haben nach SCANCO Medical AG (2024) das Format `.ISQ`. Dieses Format fällt speziell auf die Geräte der Firma SCANCO zurück. Wie das vorherige Kapitel 2.2.1 bereits eingeführt hat, ist dieser Dateityp für eine weitere Bearbeitung nicht geeignet. Unter anderem wegen ihrer Größe. Rösch und Kunzelmann (2018) haben hierfür ein Paket entwickelt. Dieses konvertiert ein `.ISQ` Format in ein `.mhd` Format. Bei einer `.mhd` Datei handelt es sich um ein Metafile, dass auf die eigentliche Datei verweist. Folgender Ausschnitt zeigt die Verwendung des Pakets.

```
python3 isq_to_mhd.py <quelle> <ziel>
```

Diese Meta-Datei kann genutzt werden, um interessante Informationen über das Bild zu erlangen. Wird dieses Kommando ausgeführt, so erstellt das Skript `isq_to_mhd` ein Metafile, das detailliert Daten über die Datei enthält. Ein Ausschnitt dieses Metafiles liefer das Listing 2.1

```
1 ObjectType = Image
2 NDims = 3
3 CenterOfRotation = 0 0 0
4 ElementSpacing = 0.02 0.02 0.02
5 DimSize = 1024 1024 517
6 ElementType = MET_SHORT
7 ElementDataFile = P01A-C0005278.ISQ
```

2.1.Listing.: *Ausschnitt des Inhaltes einer MHD-Datei*

In der Datei sind Informationen über die Ausprägung, Art und Größe der Datei zu finden. Besonders interessant sind die Punkte `DimSize und ElementType`. Über diese

Parameter lässt sich die Größe eines Bildes berechnen. Burger und Burge (2009, Seite 10-11) erklärt, das ein Bild in Zellen aufgeteilt ist, welche Informationen enthalten. Diese Zellen sind im zweidimensionalen Raum als Pixel bekannt. Betrachtet man jedoch ein, wie im Falle der Zahnklinik an der LMU dreidimensionales Bild, so spricht man nicht mehr von einem Pixel, sondern von einem Voxel. Ein Voxel ist demnach das dreidimensionale Äquivalent zu einem Pixel. Burger und Burge (2009, Seite 10-11) beschreibt weiter das jeder diese Zellen ein binäres Wort der Länge 2^k ist. Die Basis 2 ergibt sich durch das binäre Wort, wo hingegen für k gilt: $k \in \mathbb{N}$. Um für den konkreten Fall aus Listing 2.1 das entsprechenden k zu ermitteln, muss der `ElementType` näher betrachtet werden. `MET_SHORT` steht hierbei für Signed short, was eine Größe von 16 Bit entspricht. Damit ergibt sich für die Länge k ein Wert von 4. So können nach Burger und Burge (2009, Seite 10-11) folgende Gleichungen festgehalten werden.

$$\begin{aligned} 1024 \cdot 1024 \cdot 517 &= 542,113,792 \text{ Voxel} \\ 542,113,792 \text{ Voxel} \cdot 2 \text{ Byte/Voxel} &= 1,084,227,584 \text{ Byte} \\ 1,084,227,584 / 1,000,000,000 &= 1.0842 \text{ GB} \end{aligned} \tag{2.1}$$

Die erste Gleichung bestimmt die Gesamtzahl aller Voxel in einem Bild. Gleichung 2 ermittelt die Größe des Bildes in der Einheit Byte. Die letzte Zeile nimmt eine Umrechnung von Byte nach Gigabyte (GB) vor.

Durch die Gleichungen in 2.1 wird klar, das eine CT-Aufnahme des Typs `.ISQ` direkt nach seiner Aufnahme über einen GB groß ist. Laut Poliklinik (2024) ist dies ein zu großes Format. Es stellt sich also die spannende Frage, wie solch eine Datei komprimiert werden kann, ohne das es Verluste in der Qualität gibt. Dr. Elisa Walter hat hierfür eine Lösung entwickelt. QUELLE Betrachtet man den `ElementType` genauer, so fällt auf, dass es noch weiter Typen gibt, die durch eine geringere Länge k deutlich weniger Speicher benötigen. Durch Anwendung simpler Statistik lässt sich herauslesen, das die 2^4 Byte je Element nicht ausgenutzt werden. Als Werkzeug für die Betrachtung einer solchen Statistik kann das Histogramm eines Bildes genutzt werden. Laut Jähne (2024, Seite 249) ist ein Histogramm die Häufigkeitsverteilung der Grauwerte. Diese zeigt Grafisch die unterschiedlichen Grauwerte (X-Achse) zu ihren Häufigkeiten im Bild (Y-Achse). Jähne (2024, Seite 249) macht deutlich, dass das Histogramm jedoch kein AUfschluss über die räumliche Verteilung der Pixel oder Voxel liefert. Werden einige der Argumente nicht verwenden, so kann der `ElementTyp` verkleinert werden

2.3. Bildbearbeitung

Nachdem ein CT erzeugt und gegebenenfalls komprimiert wurde, verarbeitet werden. Für eine weitere Verarbeitung der Micro-CT Bilder bietet das Pipeline-Modell von Handels (2000, Seite 50) eine gute Richtlinie. Er beschreibt mit dieser Visualisierungs-Pipeline Schritte, die bei der Bearbeitung und dreidimensionalen Darstellung von CT-Aufnahmen notwendig sind (vgl. Handels 2000, Seite 50). Die ersten Schritte, *Bildvorverarbeitung* und *Segmentierung*, sind von besonderem Interesse. Dieser Abschnitt orientiert sich an dieser Unterteilung und nimmt sie als Vorbild. Daraus ergeben sich die Abschnitte 2.3.1 Filter und 2.3.2 Segmentierung.

2.3.1. Filter

CT-Aufnahmen rauschen, dies ist ein Fakt und liegt in der Natur einer Röntgenaufnahme. Dies beschreiben auch Diwakar und Kumar (2018, Kapitel 3) in ihrem Paper über CT-Bildrauschen und Entrauschen. Dabei liegt die Ursache des Rauschens nicht an einer Stelle sondern ist auf viele Quellen zurückzuführen. Einen gute Einteilung dieser Quellen liefern ebenfalls Diwakar und Kumar (2018, Kapitel 3). Sie teilen die Rauschquellen auf in *Random noise, Statistical noise, Electronic noise und roundoff noise*.

Unter dem Rauschen eines Bildes versteht man die Streuung der Pixelwerte im Bild. Für eine Segmentierung des Bildes ist dieses Verhalten unerwünscht und führt zu schlechten Ergebnissen (vgl. Handels 2000, Seite 51). Die Bildvorverarbeitung oder auch Filter gennant, hat die Aufgabe dieses Rauschen so gut wie möglich zu redzieren. Hierzu gibt es diverse Möglichkeiten.

Mit Blick auf die folgenden Kapitel sind für diese Arbeit vorallem die lokalen Operatoren relevant. Die lokalen Operatoren sind charakteristisch für die Betrachtung der lokalen Nachbarschaft. Jeder Pixel betrachtet also seine Umgebung und führt auf Basis darauf eine Berechnung des jeweils betrachteten Pixels durch (vgl. Handels 2000, Seite 52).

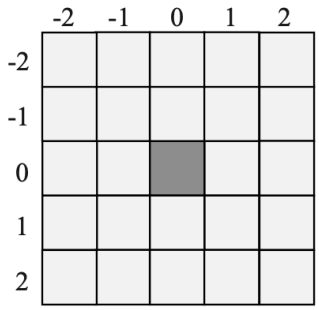

Abbildung 2.4.: *Maske eines lokalen Operators nach Handels (2000, Seite 52)*

Für die konkrete Betrachtung der Nachbarschaft eines Pixels empfiehlt Handels (2000, Seite 52) eine konkrete Maske (Ausschnitt) herranzuziehen, die mit einer Matrix interpretiert werdenen kann und die Nachbarschafft eines Pixels abdeckt. Abbildung 2.4 zeigt eine

sollche Maske und soll das Verfahren so verdeutlichen. Der grau hinterlegte Mittelpunkt ist das aktuell betrachtete Pixel. die Felder um die Mitte herum die Nachbaren. Es fällt jedoch auf, dass durch dieses Schema nicht jede mögliche Ausprägung einer Maske in frage kommt. Um ein Mittelpunkt und damit einen aktuellen Pixel betrachtetn zu können, bedarf es einer ungeraden Seitenanzahl. Diese eingränzung lässt sich wie folgt generisch fassen.

$$M_{(2_m+1)x(2_m+1)} = \begin{bmatrix} k_{11} & k_{12} & k_{1n} \\ k_{21} & x & k_{2n} \\ k_{n1} & k_{n2} & k_{nn} \end{bmatrix} \qquad m \in \mathbb{N} \tag{2.2}$$

Die Gleichung 2.2 beschreibt die mögliche Ausprägung eines lokalen Operators als Matrix. Dabei sei $m \in \mathbb{N}$. Die Variable x beschreibt das aktuell betrachtete Pixel, während k die Nachbarn illustrieren soll. Durch die Gleichung ist auch zu erkennen, das die Maske des lokalen Operators beliebig groß werden kann. Eine hohe Ordnung der Operatormatrix ist jedoch nicht immer von Vorteil, sodass es letzten Endes auf den Anwendungsfall ankommt.

Mit der Technik der lokalen Operatoren können nun unterschiedliche Arten angwenden werden. Handels (2000, Seite 54 - 55) unterscheidet hier in Glättungsfilter, Mittelwertfilter, Medianfilter, Gaußfilter und Binomialfilter. Alle dieser Filter bedienen sich einer Operatormaske um auf Basis der Nachbarelementen ein Statistischen Wert für den Bildpunkt zu erhalten. Um einen genaueren Einblick in alle Filter zu erlangen, sei an dieser Stell auf Handels (2000, Seite 54 - 55) verwiesen.

Wie zu Anfang dieses Kapitels beschrieben, ist eine Bildvorvearbeitung (Filterung) für eine gute Segmnetierung des Bildes unerlässlich. So kommt es das auch in der Visualisierungs-Pipelin nach Handels (2000, Seite 50) der zweite Schritt bereits die Segmentierung einführt. Warum dies so ein wichtiger Bestandteil der Bildanalyse ist und welche Methoden sich hier bieten, erläutert das folgende Kapitel.

2.3.2. Segmentierung

Die Bildsegmentierung oder Bildaufteilung ist ein wichtiges Teilgebiet der Bildverarbeitung und beschäfftigt sich mit der Bildanalyse. Ihr Ziel ist es, detailirtere beschreibende Bilder aus dem vorliegenden Orginalbild zu berechnen. Dies kann im Falle eines CTs in der Zahnklinik an der LMU München die hervorgehobene Darstelllung der Zahnsubstanzen Schmelz und Dentin sein. (vgl. T. Lehmann u. a. 2013, Seite 359). Konkret teilt ein Segmentierungsverfahren also ein Bild in Teilbereiche auf. Dabei sind die Teilbereich in sich bemerkenswert homogen. Ramesh u. a. (2021, Seite 1) beschreibt, dass der Prozess der Segmentierung zur Gewinnung wichtiger Informationen dient wie zum Beispiel die Zahnkaries Ausbreitung. So kommt es, dass Handels (2000, Seite 50) in seiner Visualisierungs-Pipelin die Segmentierung als zweiten Schritt und damit als zentrales Problem darstellt.

Handels (2000, Seite 95) und T. Lehmann u. a. (2013, Seite 360) beschreiben beide, dass die Bildsegmentierung eines CTs für eine gute und eindeutige Ärzliche Diagnose nicht mehr wegzudenken ist. Warum dem so ist, verdeutlicht die Abbildung 2.5.

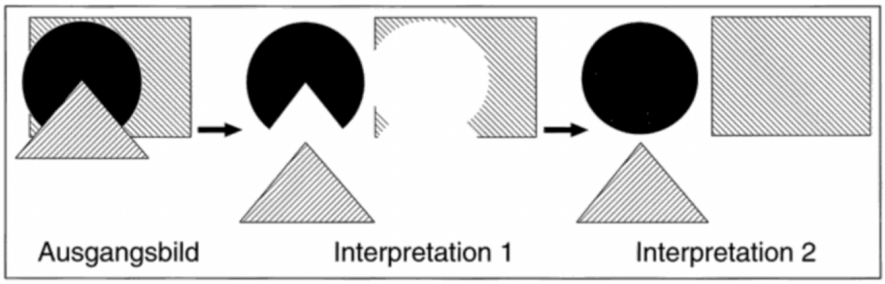

Abbildung 2.5.: *Interpretation einer CT-Aufnahme nach T. Lehmann u. a. (2013, Seite 360)*

Zu erkennen ist das orginale Bild (Ausgangslage) und mögliche Interpretationsschritte. T. Lehmann u. a. (2013, Seite 360) verdeutlicht mit dieser Abbildung 2.5, dass mittels orginalem Bild die einzig mögliche Interpretation die erste ist. Auch wenn die zweite Interpretation die deutlich logischere ist, kann diese ohne weitere Forschung nicht bewiesen werden, so T. Lehmann u. a. (2013, Seite 360). Für die Schlussvolgerung von Interpretation 1 auf Interpretation 2 ist eine Segmentierung des Bildes notwending. Erst damit lässt sich zeigen, wie die Strukturen wirklich aussehen. So beweist dieses Bild, dass die Segmentierung ein wesentlicher Teil der Bildanalyse ist.

Um ein Bild zu segmentieren gibt es unzählige Möglichkeiten. Für die Auswhal eines Verfahrens spielt unter Anderem der Anwendungsbereich eine wichtige Rolle. Die Verfahren, die in dieser Arbeit von Wichtigkeit sind, sind die Schwellwertverfahren (vgl. T. Lehmann u. a. 2013, Seite 361).

Schwelltwertverfahren (engl.: thresholding) gehören zu den Standardwerkzeugen einer Segmentierung, sodass sie die Basis vieler weiterer Verfahren legen. Bei einer Schwellwertbasierten Segmentierung werden die Pixel eines Bildes anhand von Schwellwerten eingruppiert (vgl. Handels 2000, Seite 96). Die nachfolgende Gleichung 2.3 soll dies verdeutlichen.

$$B(x,y,z) = \begin{cases} 1, & \text{falls } t_{\text{unten}} \leq f(x,y,z) \leq t_{\text{oben}}, \\ 0, & \text{sonst.} \end{cases} \tag{2.3}$$

$B(x,y,z)$ Beschreibt einen Pixel in einem dreidimensonalen Bild, demnach ein Voxel. Liegen die Werte eines Voxels, also $f(x,y,z)$, innerhalb der beiden Schwellwerte t_{oben} und t_{unten}, dann wird eine 1 zugewiesen. Liegt der aktuell betrachtete Voxel nicht zwischen den Schwellwerten, so wird eine 0 zugewiesen.

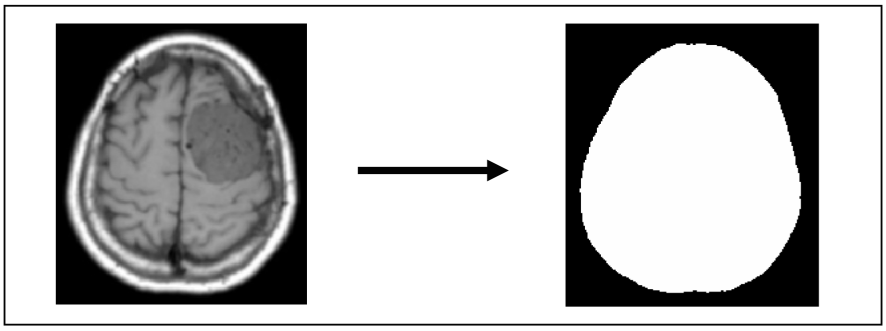

Abbildung 2.6.: *Ergebnis eines einfachen Schwellwertverfahrens nach Handels (2000, Seite 96)*

In der Abbildung 2.6 ist zu erkennen, dass nach einem einfachen Schwellwertverfahren das Bild nur noch aus zwei unterschiedlichen Graustufen besteht. Abgesehen von der Sinnhaftigkeit, ist diese einfach Segmentierung durchaus erfolgreich verlaufen. Der Grund dafür ist die gute Wahl des Schwellwertes.

Die interessanteste Frage bei den Schwellwertverfahren ist die Wahl des Schwellwertes t. Dieser entscheident zwischen einer guten und einer schlechten Segmentierung. Für die Wahl eines Schwellwertes empfielt sich der Blick auf das Bildhistogramm. Dieses gibt Aufschluss über die Grauwertverteilung eines Bildes (vgl. T. Lehmann u. a. 2013, Seite361). Verfahren, welche eine gute Schwellwertwahl gewährleistet, ohne das zu viele Informationen verloren gehen, sind die Verfahren *Otsu* und *Renyi*.

Das Verfahren nach Otsu gehört zu den Schwellwertverfahren und und bestimmt den Schwellwert t durch ein statistische Gütekriterieum. Hierzu bedient sich das Verfahren des Bildhistogrammes. Das tatsächliche Bild, und damit die räumliche Anordnung der Voxel benötigt dieser Algorthmus nicht (vgl. T. Lehmann u. a. 2013, Seite 264).

Ein solches Histogramm, welches die Grundlage für für das Verfahren nach Otsu liefert sei in Abbildung 2.7 gezeigt. Dies gibt Aufschluss über die unterschiedlichen Grauwerte und wie oft sie in einem Bild vorkommen (vgl. T. Lehmann u. a. 2013, Seite264). Für eine genauere Beschreibung eines Histogrammes, sei an dieser Stelle auf Burger und Burge (2009, Seite42) verwiesen.

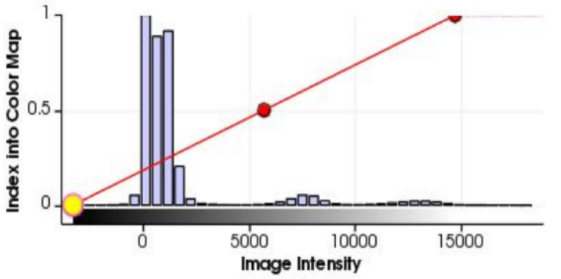

Abbildung 2.7.: *Histogramm einer Zahnaufnahme nach Hoffmann*

Das Otsu-Verfahren teilt die Grauwerte eines Bildes in verschiedenen Klassen ein, die durch Schwellwerte getrennt werden. Die Klassen können beispeilsweise mit K_0 bis K_n bezeichnet werden, wobei sich dieses konkrete Beispiel auf die Klassen K_0 und K_1 beschränkt. Otsu wählt den Schwellwert t, der die Varianz zwischen den Pixelklassen maximiert und gleichzeitig die Varianz innerhalb jeder Klasse minimiert (vgl. T. Lehmann u. a. 2013, Seite264). Matematisch lässt sich dies wie folgt ausdrücken.

$$t = \max\ (\sigma_{zw}^2 / \sigma_{in}^2) \tag{2.4}$$

σ_{zw} bildet die Varianz zwischen den beiden Klassen K_0 und K_1 und wird gebildet aus der Wahrscheinlichkeiten mit denen jeder einzelne Grauwert auftritt. σ_{in} hingegen, ist die Varianz innerhalb einer Klasse und entsteht durch die Addition der Varianzen der einzelnen Klassen. Der Schwellwert t ist nun der, für den das Verhältnis maximal wird (vgl. T. Lehmann u. a. 2013, Seite264).

Laut T. Lehmann u. a. (2013, Seite264) fällt auf, dass dieses Verfahren vorallem bei bimodalen Bildern zum Einsatz kommt. Ein Bild ist bimodal, wenn es zwei lokale Maxima aufweist. Das Otsu-Verfahren ist jedoch nicht auf eine Bimodalität beschränkt und kann auf beliebig viele Klassen erweitert werden (vgl. T. Lehmann u. a. 2013, Seite264).

Neben unzähligen Segmentierungstechniken, ist eine für diese Arbeits von ganz besonderer Bedeutung. Diese Technik wurde speziell zum Segementieren der Micro-CT bilder des Zahnklinikums an der LMU in München entwickelt und bildet die Basis dieser Arbeit. Konkret ist damit das Hoffmann-Verfahren gemeint (vgl. Hoffmann 2020).

Das Verfahren nach Renyi ist ein weiteres Verfahre, das im Laufe dieser Arbeit noch eine wichtige Rolle spielt. Wie bereits Beschrieben gehört es ebenfalls zu der Gruppe der Schwellwertverfahren und genneriert demnach Schwellwert t. Wie auch das Verfahren nach Otsu, benötigt Renyi keine Information über die räumliche Anordnung der Bilder. Es genügt das Bildhistogramm. QUELLE

TODO

2.4. Verwandte Arbeit

Wie bereits in der Einleitung dieser Arbeit klar wurde verfügt die Poliklinik für Zahnerhaltung und Parodontologie des LMU-Klinikums München über einen breiten Schatz an Bilddaten. Im Rahmen einer Bachelorarbeit an der Hohschule für angewnadte Wissenschafften in Augsburg unterstützte Hoffmann (2020) die Verarbeitung dieser Bilddaten mit Methoden der 3D-Bildverarbeitung. Konkret sollte die Arbeit die Kariesklassifizierung unterstützen. Hierzu entwickelte er ein Verfahren, das auf Basis adaptiver Schwellwertverfahren die Zahnsubstanzen Schmelz und Dentin aus dem Orginalbild herrauslöst. Konkret kann diese Segmentierung mir zwei verschiedenen Verfahren durchgeführt werden.

Durch die Segmentbetrachtung der Beiden Zahnhauptteile Schmelz und Dentin konnte Hoffmann (2020) eine gute Hilfe für die Befundung kariöse Stellen liefern. Ein Ergebnis aus der Arbeit von Hoffman sei in Abbildung 2.8 gezeigt. Hoffmann (2020) entwickelte hierfür ein Prototypisches Verfahren, mit dem es gelang ca. 250 Datensätze der Zahnklinik automatisch aufzubereiten.

Abbildung 2.8.: *Reproduzierte Ergebnisansicht eines CTs nach Bearbeitung mit dem Verfahren nach Hoffmann (2020, Seite 56)*

Hoffmann (2020) entwickelte hierfür ein Prototypisches Verfahren, mit dem es gelang ca. 250 Datensätze der Zahnklinik automatisch aufzubereiten. Hoffmann (2020) beschreibt weiter, dass dieses Verfahren bis zu einem gewissen Vortschritt des Karies durchgeführt werden konnte, da der Algorithmus diesbezüglich seine Grenzen hat. Für die Spätere Darstellung der Ergebnisse kann eine überlappende Ansicht in einer Visualisierungssoftware verwendet werden.

So ergibt sich die Situation, dass der Algorithmus ein gutes Ergebnis liefert, jedoch nicht benutzerfreundlich zu bedienen ist. Für das Starten und Visualisueren des Verfahrens sind aufwendige Befehle über das Terminal zu tippen (vgl. Hoffmann 2020, Seite 53). Genau an dieser Stelle soll die vorliegende Arbeit anknüpfen und das Verfahren nach Hoffmann (2020) so interaktiv und benutzerfreundlich gestallten.

Für eine interaktive Verarbeitung von 3D Bilddaten bieten sich einige Möglichkeiten. Die wohl beste Lösung liefer 3D Slicer. Warum die Wahl auf diese Plattform viel und welche Vorteile daraus entstehen wird im folgenden Abschnitt 2.5 erläutert.

2.5. Interaktive Bildbearbeitung mit 3D Slicer

3D Slicer ist eine Open Source Plattform, die speziel für die Verarbeitung für Bilddaten im medizinischen Kontext eingesetzt wird. Dabei wird sie von einer aktiven Community regelmäßig gewartet und weiterentwickelt (vgl. 3D Slicer Community 2024a), (vgl. Fedorov u. a. 2012). Für Slicer gibt es offizell keine Nutzungsbeschränkung. Jedoch sei auch gesagt, dass 3D Slicer nicht für die klinische Nutzung zugelassen ist. Fedorov u. a. (2012) macht deutlich, das 3D Slicer ausschließlich für die Forschung gedacht ist. Um einen ersten Überblick über die Komponenten von Slicer zu erlangen, soll die Abbildung 2.9 betrachtet weden.

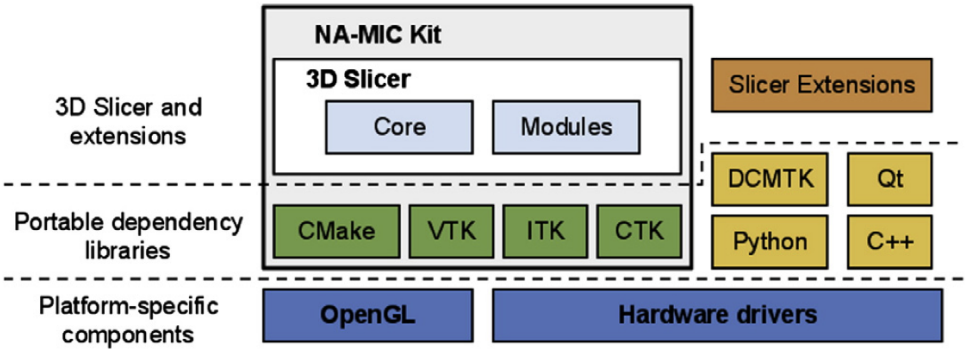

Abbildung 2.9.: *3D Slicer Ökosystem nach Fedorov u. a. (2012, Seite 1326)*

Fedorov u. a. (2012, Seite 1326) Teilt mit der Abbildung 2.9 die Plattform in drei Schichten auf. Auf der obersten wird klar, dass 3D Slicer aus der Kernanwendung und den Installierbaren Modulen besteht. Neben den bereits vorhandenen Modulen können von externen Entwicklern Module über die Slicer Extentions entwickelt und bereitgestellt weden. Um eine Weiterentwicklung möglich zu machen hat Slicer eine Reihe von Abhängigkeiten, die jedoch portabel gehalten werden. Auf der untersten Schicht sind die Platformspezifischen Anforderungen zu sehen, die Slicer erfüllen soll.

So kommt es, dass das 3D Slicer Ökosystem sich durch einige Kriterien besonders auszeichnet. Die wohl wichtigsten seien hier Stichpunktartig genannt (vgl. 3D Slicer Community 2024a), (vgl. Fedorov u. a. 2012).

- Kostenfreie Software
- Plugin Infrastruktur durch den Extention Manager
- Ausführen von Skripten in der integriertetn Python Konsole
- Verarbeitung von medizinischen Bilddaten von Kopf bis Fuß
- Interaktive Benutzerschnittstelle

3D Slicer hat für alle diese Punkte jeweils eine Lösung entwickelt, wobei der erste Punkt durch die Open Source Philosiphie schon gegeben ist. Die folgenden Abschnitte decken diese Lösungen ab und bilden so eine erste Grundlage für die Entwicklung mit 3D Slicer(vgl. 3D Slicer Community 2024a), (vgl. Fedorov u. a. 2012).

2.5.1. Extension Manager und Plugin Infrastruktur

Der wohl bedeutenste Punkt ist die Plugin Infrastruktur, welche Slicer von sich aus mitbringt. Um dieses Konzept genauer zu beläuchten Teilt man die Platform am besten in zwei Teile auf. Die Kernanwendung und die Module, welcher jeder User personalisiert installieren oder deinstallieren kann. Diese Module werden als *Slicer lodabel module* bezeichnet (vgl. Fedorov u. a. 2012, Seite 1332). Slicer realisiert die Struktur durch den Extention Manager, welcher durchaus vergleichbar ist mit dem AppStore. Über diesen können bequem und mit wenig Klicks die gewünschten Erweiterungen in das Kernsystem installiert werden.

Neben der Möglichkeit Module zu installieren bietet Slicer noch die Möglichkeit eigenen Module zu bauen und Sie im Extention Manager zu veröffentlichen. Diese werden als *Slicer Extension Module (SEM)* bezeichnet. Hierzu verfolgt Slicer den Ansatz, dass jeder Entwickler eines Moduls selbst verantwortlich für Wartung und Weiterentwicklung ist. Auch nachdem ein Paket veröffentlicht wurde (3D Slicer Community 2024a).

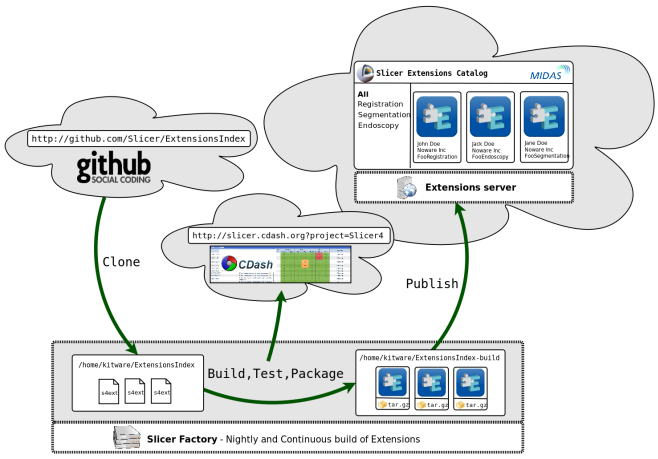

Abbildung 2.10.: *Funktionsweise der Plugin Infrastruktur von 3D Slicer nach 3D Slicer Community (2024b)*

Slicer realisiert dies, indem die Plattform über ein zusätzliches Reporsitory verfühgt, dass sich *ExtensionIndex* nennt. Dieses öffentliche Repository ist eine Auflistung aller Slicer Extentions. Die Auflistung erfolgt durch eine Reihe an *JSON*-Files, die auf die Repositories der einzelnen Entwickler verweisen. Dieser *ExtensionIndex* ist über die Slicer Factory an den Extention Server und damit auch an den Extention Managerx angebunden. Die Slicer Factory ist ein System, das aus einem Slicer Extention Repository ein lauffähiges Build erstellt, welches in den Extention Katalog eingebunden werden kann.

Ist eine Extention in dem Extention Katalog gelistet, so sorgt der Extention Manger dafür, dass die von der Slicer Factory erstellt build-Datei installiert werden kann. Abbildung **??** soll diesen Vorgang verdeutlichen (vgl. 3D Slicer Community 2024a).

Die Kernanwendung von 3D Slicer folgt einem Softwarepattern, dass sich *Model View Controler (MVC)* nennt. Bei der Erstellung einer SEM soll dieser Absatzt ebenfalls gepflegt werden. Eine High Level Betrachtung der Softwarearchitektur von 3D Slicer bietet Fedorov u. a. 2012, Seite 1332 mit der Abbildung 2.11.

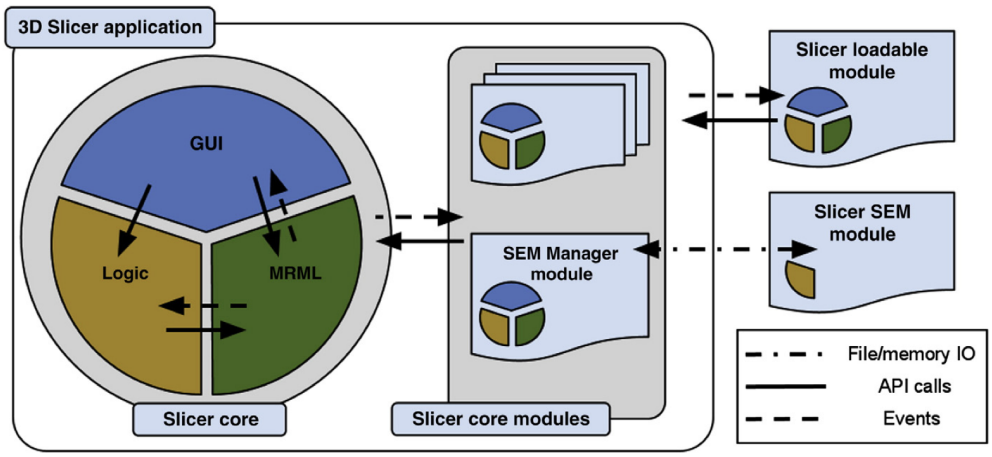

Abbildung 2.11.: *3D Slicer High Level Architektur nach Fedorov u. a. (2012, Seite 1326)*

Das Zusammenspiel zwischen *GUI, MRML und Logic* bilden das MVC-Pattern in der Kernanwendung. Das indentische Pattern spiegelt sich auch in den einzelnen Modulen von Slicer wieder. So wird sichergestellt, dass ein Softwareentwicklungsparadigma eingehalten wird, was sich *separation of concerns* nennt. Die Kapselung von zusammengehöriger Logik. Bei der Erstellung einer eigenen Extention ist die Idee, dass nur die Logic implementiert werden muss und die komplexe Architektur von Slicer erstmal nicht relevant ist.

Jedoch bietet sich in Slicer nicht nur die Möglichkeit eigenen Erweiterungen zu erstellen. Es lässt sich hierfür auch die integrierte Python Konsole nutzen.

2.5.2. Python Umgebung

3D Slicer bringt eine integrierte Python Konsole mit, über die mit der Datenstruktur interagiert werden kann. So ist es möglich Python Skripte direkt in der Konsole auszuführen. Um dies zu realisieren bringt Slicer mit der Installation im jeweiligen Betriebssystem eine eigenen Python Umgebung mit. Dieses sieht wie folgt aus.

```
./Slicer/bin/PythonSlicer
```

Diese Python Umgebung verfügt über alle notwendigen Abhängigkeiten und Pakete. Bei der Entwicklung eines SEM kann dann auf die Pip-Pakete in der integrierten Python Umgebung zurückgegriffen werden. So kommt es das für eine Entwicklung mit Slicer keine eigenen Python Umgebung auf der lokalen Maschine installiert sein muss. Slicer bringt hier alls mit.

Für den letzten charakteristischen Punkt von Slicer aus Kapitel 2.5 führt der nächste Abschnitt in die durchaus komplexe Datenstruktur MRML ein, die bei einer Entwicklung mit Slicer unausweichlich zu berücksichtigen ist.

2.5.3. MRML Datenstruktur

Die *Medical Reality Modeling Language (MRML, gesprochen "MURml")* ist ein Datenmodell das dafür entwickelt wurde alle möglichen Bilddaten zu visualisieren und zu speichen, die für einen klinischen Zweck Einsatz finden (vgl. 3D Slicer Community 2024a). Laut der 3D Slicer Community (2024a) wurde die MRML-Datenstruktur völlig unabhängig von der Slicer Kernanwedung entwickelt. Dies ermöglicht ein portieren der Datenstruktur auf andere Softwareapplikationen. Da Slicer die einzig große Plattform ist, die diese Datenstruktur nutzt, wird der Quellcode für MRML im Reporsitory von 3D Slicher gewartet und weiterentwickelt, so 3D Slicer Community (2024a). Durch den Artikel von Fedorov u. a. (2012, Seite 1327) wird klar, dass MRML mehr ist also nur eine Datenstruktur ist. Sie beschreiben MRML als Szenenorganisator von Bilder, Anotationen, Layouts und Anwendungsstaten.

Fedorov u. a. (2012, Seite 1331) beschreiben die MRML-Datenstruktur als Schlüsselkomponenten innerhalb von 3D Slicer. Dies ist auf die Softwarearchitektur von Slicer zurückzuführen, die in Abbildung 2.11 beschrieben wurde. Die Kernanwendung von Slicer arbeitet nach dem MVC-Pattern. MRML übernimmt hier den Teil des *Models (M)* und bildet damit den Grundstein der Anwendung (vgl. Fedorov u. a. 2012, Seite 1332).

Die 3D Slicer Community (2024a) und der Artikel von Fedorov u. a. (2012, Seite 1327) beschreibt MRML als XML-Format. Wird also eine MRML-Szenen gespeichert, so folgt eine Speicherung als .mrml-Datei und damit unter der Haube als XML-Datei. Dabei wird laut 3D Slicer Community (2024a) nur eine Referenz auf das Bild gespeichert. Die zu bearbeitende Aufnahme selbst wird nicht inerhalb einer MRML-Datei abgespeichert.

MRML zeichnet sich vorallem dadurch aus, dass es eine Vielzahl an Dateiformaten akzeptiert. Alle Formate, die für einen klinischen zweck verarbeitet werden, können durch MRML untersützt werden. Um dies zu gewährleisten, ist die MRML Szene in

sogenannte *nodes* aufgeteilt. Die basis Node-Typen folgen der 3D Slicer Community (2024a) und sind in der folgenden Aufzählung zu sehen.

- Data nodes
- Display nodes
- Storage nodes
- View nodes
- Plot nodes
- Subject hierarchy node
- Sequence node
- Parameter node

Wird also ein Bild in eine MRML-Szene geladen, so speichert Slicer die unterschiedlichen Eigenschaften eines Bildes in unterschiedlichen nodes. So werden Beispielsweise basis Eigenschaften einer Probe im *Data node* gespeichert, wo hingegen ein *Storage node* beschreibt wie eine *Data node* in einer Datei gespeichert wird. In *Display node* werden die Eigenschaften zur Darstellung eines Bildes hinterlegt. Der Hintergrund für die Speicherung von Probendaten in unterschiedllichen nodes ist, dass beispielsweise das selbe Bild in unterschiedlichen Formaten vorliegt oder ein und das selbe bild auf zwei unterschiedliche Arten visualisiert werden soll. So kann sich Beispielsweise eine Struktur wie in Abbildung 2.12 ergeben.

Abbildung 2.12.: *3D Slicer High Level Architektur nach 3D Slicer Community (2024a)*

Die Informationen in einem Bild werden also über dies Typen aufgeteilt und je nach Sinn abgespeichert. Möchte man demnach auf die bestimmte Informationen in einer Probe zugreifen. So kann diese Information über den Aufruf bestimmter Methoden erfolgen

```
1 # data node - vtkMRMLVolumeNode
2 currentVolume.GetImageData()
3 # storage node - vtkMRMLStorableNode
4 currentVolume.GetStorageNode()
5 # display node - vtkMRMLDisplayableNode
6 currentVolume.GetDisplayNode()
```

2.2.Listing.: *Auslesen der Informationen aus den verschiedenen nodes*

Wie die Kommentaren in Listing 2.2 bereits zeigen, gibt es noch eine besonderheit von MRML. Damit ein Verwaltung alle Dateiformate möglich ist, bedient sich MRML einiger Tools, die sich bereits etabliert haben. Die wichtigsten sind das VTK Development Team (2024) und das ITK Development Team (2024). Dies beiden Tools sind echte Rießen in ihrere Branche. MRML nutzt diese um einige Dateiformate zu lessen und zu schreiben.

Durch das Betrachten der MRML-Szene wird klar, dass Slicer hierdurch viele Möglichkeiten bietet. Speziell für die effiziente Speicherung der Proben in einer Szene durch die unterschiedlichen node Typen. Ein besnoderer node, der gleichzeitig auch die Brücker zu der interaktiven Benutzerschnittstelle von Slicer baut, ist der *Parameternode*. Warum

dieser eine zentrale Rolle spielt und wie Slicer die Schnittstelle grundsätzlich gestalltet, soll in Kapitel 2.5.4 Benutzerschnitstelle diskutiert werden.

2.5.4. Benutzerschnittstelle

Für das erstellen eines Benutzerschnittstelle (engl. *User Interface (UI)*), das für eine Slicer Extention nötig ist, nutzt 3D Slicer den Qt-Designer (vgl. Qt Development Team 2024). Die Integration des Qt-Designers als Applikation in eine andere Applikation funktioniert aufgrund der Plattformintegrität, die der Designer mitbringt (vgl. Qt Development Team 2024). Dieser bietet so die Möglichkeit die benötigten Widgets über ein interaktives User Interface zu bauen. Für dieses UI-Widget gibt es einen Gegenspieler im Quelltext des Programmes, welcher als *ParameterNode* bekannt ist. Der *ParameterNode* ist laut 3D Slicer Community (2024a) eine leichte Variante eines MRML-Node um Parametereinstellungen zu speichern. Durch das Zusammenspiel zwischen UI-Widget und *ParameterNode* wird die UI automatisch aktualisiert, wenn sich das Programm ändert und umgehert 3D Slicer Community (2024a).

Das Erstellen der Verknüpfung zwischen UI-Widget und *ParameterNode* erfolgt über die dynamische Eigenschaft `SlicerParameterName`, die direkt in der Komponentenansicht im Qt-Designer einstellbar ist. Die Abbidlung 2.13 soll diesen Vorgang verdeutlichen. Diese Verknüpfung lässt sich laut 3D Slicer Community (2024a) auch via Programmcode setzten.

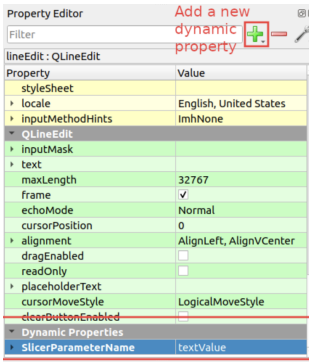

Abbildung 2.13.: *Dynamische Eigenschaft einer Komponenten im Qt-Designer nach 3D Slicer Community (2024a)*

```
widget.setProperty('SlicerParameterName', 'parameterName')
```

Über das Objekt *widget* kann die eigenschaft einer Komponente gesetzt werden, ohne dass sie im Designer berührt werden muss.

Mit dem Ende dieses Abschnittes wurden alle wichtigen Bestandteile von 3D Slicer abgedeckt und diskutiert, sowie alle weiteren Domänen eingeführt. So bleibt nun die Frage nach dem Sinn dieser Arbeit. das Kapitel 3 soll hier Klarheit liefern und die konkrete Fragestellung ausarbeiten.

3. Fragestellung

Dieses Kapitel beleuchten die zentrale Frage, welche mit Hilfe der Ergebnisse dieser Arbeit beantwortet werden sollen. Dabei kann die Fragestellung in erster Linie als Ergebnis der theoretischen Grundlagen Kapitel 2 interpretiert werden.

Die vorliegende Arbeit befasst sich mit der Erstellung einer Slicer-Extension, die ein spezifisches Segmentierungsverfahren auf Basis der Methode nach Hoffmann integriert. Die Grundlagen für diese Methode bilden die theoretischen Grundlagen.

Die Methode nach Hoffmann nutzt bestehende Segmentierungsmethoden und wurde speziell für die Anforderungen der Zahnklinik in München entwickelt. Sie ist prototypisch implementiert und zeigt in ihrer Funktionalität vielversprechende Ergebnisse. Allerdings hat das aktuelle Verfahren eine wesentliche Einschränkung: Es muss über das Terminal ausgeführt werden. Dies erschwert die Anwendung in der klinischen Praxis und reduziert die Benutzerfreundlichkeit erheblich.

Als interaktive Lösung bietet sich die Software 3D Slicer an, da sie bereits in der Zahnklinik eingesetzt wird und über eine flexible Infrastruktur für die Integration von Erweiterungen verfügt. Diese Eigenschaften machen Slicer zu einer idealen Plattform, um die Methode nach Hoffmann in einer benutzerfreundlichen Form bereitzustellen.

Die zentrale Frage, die so aus den Grundlagen abgeleitet werden kann, ist:

1. Kann das Verfahren nach Hoffmann als Extension in Slicer implementiert werden?
2. Ist es möglich, die Integration so zu gestalten, dass der zugrunde liegende Algorithmus problemlos austauschbar ist oder zusätzliche Algorithmen eingebaut werden können?
3. Welche Erfahrungen lassen sich bei der Entwicklung einer Slicer-Extension sammeln, und welche Herausforderungen treten dabei auf?

Im folgenden Kapitel wird dargestellt, wie diese Fragestellung methodisch bearbeitet wurde. Es wird gezeigt, wie das Problem strukturiert in Teilaufgaben zerlegt wurde und welche Schritte zur Lösung unternommen wurden.

4. Methodik

Wie das vorherige Kapitel bereits eingeleitet hat, soll es hier um das *Wie* gehen. Es wird also aufgezeigt, welches methodische Herangehen an die Fragestellung verfolgt wurden, um ein aussagekräftiges Ergebnis zu erzielen. Dabei wurde mit einer umfangreichen Anforderungsanalyse gestartet, welche die Domäne und die Ausgangslage klären soll. Sind diese Konzepte klar, so wird die endgültige Problemstellung in mehrere kleine Teilaufgaben zerlegt. Für jede dieser Teilaufgaben wurde dann eine Recherche zum Stand der Technik durchgeführt, um bereits existierenden Lösungen ausfindig zu machen. Sofern es für eine Teilaufgabe noch keine Lösung gibt, werden anschließend konkrete Lösungsansätzen für die einzelnen Teilprobleme erarbeitet. Sollte für eine Aufgabe mehrere Ansätze existieren, so werden diese im letzten Abschnitt miteinander verglichen und ein passender Ansatz gewählt.

Bei der Durchführung dieser Schritte zum Erreichen eines Ergebnisses, soll der in der Softwareentwicklung allgemeine bekannte Ansatz *make it run, make it right, make it fast* verfolgt werden. Dieser beschreibt, dass zunächst dafür gesorgt werden soll, dass ein Problem überhaupt gelöst wird, bevor man viel Zeit in ein Refactoing oder eine Optimierung steckt.

4.1. Anforderungsanalyse

Nach genauerem Betrachten der Fragestellung aus Kapitel 3 wird klar, dass im Rahmen dieser vorliegenden Arbeit eine Extension für die Plattform 3D Slicer entwickelt werden soll. Diese Erweiterung beinhaltet das Segmentierungsverfahren nach Hoffmann (vgl. Hoffmann 2020), wie es in Kapitel 2.4 beschrieben wurde. Das Verfahren segmentiert Micro-CT Aufnahmen der Zahnklinik in München und wird zu Forschungszwecke eingesetzt. Da Ärzte keine Softwareentwickler sind, ist es wichtig, dass das Verfahren eine UI erhält die eingängig und übersichtlich ist. Außerdem ist eine stabile Anwendung gefragt, die sich gut in die Kernanwendung von 3D Slicer einfügt. Für einen Überblick über die wichtigsten Eigenschaften von 3D Slicer sei auf das Kapitel 2.5 verwiesen.

Die Extension selber soll neben einer Einzelbildbearbeitung auch einen Batch-Prozess ermöglichen. So können Beispielsweise Parameter an einem Bild erprobt werden und

diese anschließend in eine Batchprozess für viele Bilder überführt werden. Außerdem soll es möglich sein, verschiedenen Segmentierungsverfahren, die in Hoffmann vorgesehen sind, auch in der Extention auszuwählen.

Ein wichtiger Softwaretechnischer Anspruch an die Extension ist die Erweiterbarkeit. Es soll ohne große Mühen möglich sein, ein weiteres Verfahren zu integrieren, ohne das große Anpassungen an der UI oder der Erweiterung selbst, unternommen werden müssen. Für ein solides Verständnis dieser Software soll es selbstverständlich eine Dokumentation mit Benutzerhandbuch geben. Zudem wird großer Wert auf die Qualitätssicherung gelegt, weshalb eine Reihe von Unit-Tests (Tests für einzelne Programmeinheiten) vorgesehen ist.

Um die Anforderungen an die Software besser zu verstehen und zu strukturieren, ist neben der Sammlung technischer Spezifikationen auch ein solides Verständnis für die zugrunde liegende Domäne essenziell. Die Abbildung 4.1 veranschaulicht dies durch ein UML-Domänenmodell (Unified Modeling Language), das einen visuellen Überblick über die verschiedenen Teile der Software bietet. Dazu sind auch einige Anforderungen wieder zu erkennen.

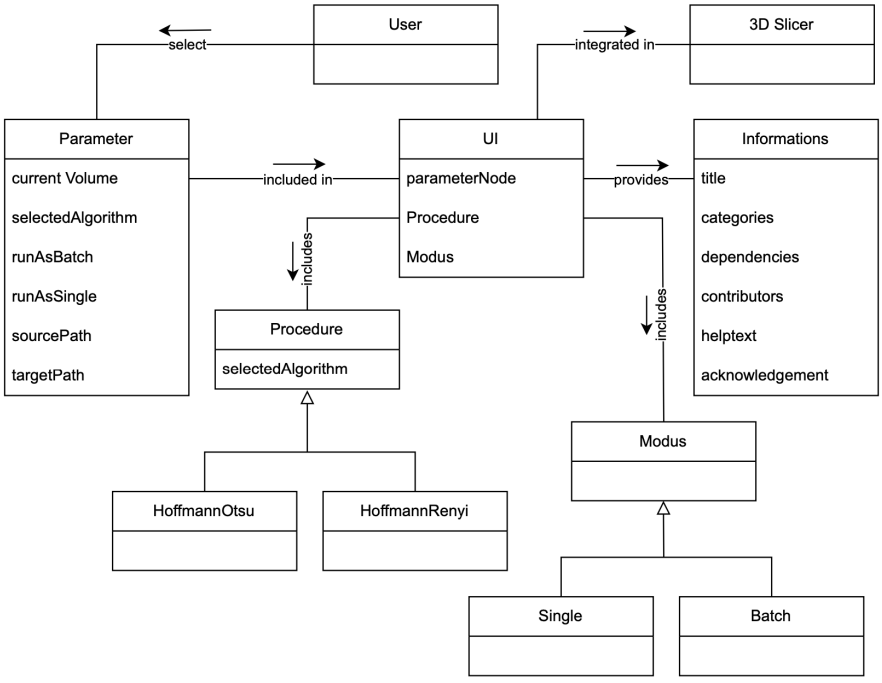

Abbildung 4.1.: *UML-Domänenmodell des gesamten Softwaresystems*

Diese doch breite Palette an Anforderungen lässt sich unmöglich auf einmal bearbeiten. Auch durch eine visuelle Darstellung kann dies nicht vereinfacht werden. Hierzu sieht

diese Arbeit eine Aufteilung in Teilprobleme vor. Der nächste Abschnitt blickt auf die herausgearbeiteten Anforderungen in diesem Kapitel und leitet daraus Teilprobleme ab.

4.2. Zerlegung in Teilprobleme

Durch die Aufteilung des Gesamtsystems in mehrere kleine Teilaufgaben wird die Software für den Entwicklungsprozess übersichtlicher. Die einzelnen Domänen können so schneller und besser verstanden werden. Es gibt viele Möglichkeiten ein Softwaresystem in kleine Teile aufzuteilen, sodass es am Ende auf den konkreten Anwendungsfall ankommt. Diese Arbeit sieht folgenden Teilaufgaben für das Gesamtsystem vor:

- **Architekturdesign:** Mithilfe von UML Diagrammen soll die Architektur dieses Systems abgebildet werden und sukzessive immer detaillierter beschrieben werden. Es soll dann verglichen werden, welche Softwarepatterns für dieses System infrage kommen. Durch die Bearbeitung dieses Teilproblems kann die Anforderung an eine flexible Architektur erfüllt werden. Anschließend kann mit der Entwicklung des UI-Designs begonnen werden.
- **UI Design:** Es soll ein Design erstellt werden, dass sich an erfolgreichen und etablierten 3D Slicer Extensions orientiert. Jedoch sollen die Wünsche des Endnutzers auch nicht zu kurz kommen. Für eine Visualisierung des Designs bedient sich diese Arbeit der Wireframes.
- **Pseudo-Extension:** Bevor der tatsächliche Algorithmus eingebunden werden kann, ist es wichtig eine funktionierende Erweiterung zu haben, die noch keine konkrete Aufgabe hat, aber funktioniert und in Slicer eingebunden werden kann.
- **Hilfsfunktionen:** Nachdem die Infrastruktur der Erweiterung steht und funktioniert, kann mit der Implementierung einiger Hilfsfunktionen begonnen werden. Hierbei handelt es sich um Methode, die nicht direkt etwas mit dem Verfahren zu tun haben, jedoch kleine Nebenaufgaben erfüllen und so unumgänglich sind. Als Beispiel sei hier das Laden von CT-Bildern in die Szene gedacht.
- **Kappselung Hoffmann:** Nachdem die leere Extension lauffähig ist und auch einige Hilfsfunktionen bereitstehen, kann mit der Paketerstellung des Hoffmann begonnen werden. Hier soll das Verfahren von einem Python Notebook in eine Bibliothek überführt werden, sodass dieses Verfahren in der Extention ausführbar ist. Die konkrete Art des Paketes ist noch nicht festgelegt.
- **Speicherung der Parameter:** Der Benutzer steuert das Verfahren über die Parameter in der UI. Für die Speicherung der Parametereinstellungen hat Slicer den Mechanismus ParameterNode entworfen. Diese wurde bereits in Abschnitt 2.5.4 erwähnt. Dieser Mechanismus ist nicht trivial, erhöht die Benutzerfreundlichkeit

des Systems aber erheblich und soll demnach auch in diese Extention Anwendung finden.

- **Single Prozess:** Sobald alle notwendigen Vorbereitungen getroffen sind, kann der Algorithmus nun eingebettet werden. Hierzu betrachtet man isoliert den Single Prozess. Auch die UI wird erst nur so weit entwickel, wie es für den einfachen Prozess nötig ist. Hierbei wird auf das erstellte Paket für das Hoffmann Verfahren und die zuvor erstellen Hilfsfunktionen zurückgegriffen.
- **Batch Prozess:** Ist das einfache Verfahren fertig implementiert und funktioniert, so kann der Batch Prozess hinzukommen. Hier bedarf es einer zusätzlichen Arbeit in der UI, da der Benutzer über das Verwenden dieser Funktion gewarnt werden muss. Der Batch Prozess bedarf nämlich erheblicher Ressourcen. Hinzukommt die Implementierung einer Fortschrittsanzeige, sodass zu erkennen ist, dass ein Hintergrundprozess läuft.
- **Dokumentation und Benutzerhandbuch:** Abschließend ist eine ausführliche Dokumentation der Architektur erwünscht, sodass zukünftige Entwickler wissen, wo sie ansetzten müssen. Hinzu kommt ein Benutzerhandbuch für eine Verwendung der Erweiterung. Das Benutzerhandbuch und die Architekturdokumentation erfolgen in einer README.md innerhalb der Extension.
- **Tests:** An letzter Stelle sollen noch Softwaretests implementiert werde, um die Richtigkeit der Extension sicherzustellen. 3D Slicer sieht hier Unittests vor, die über den Developer Modus in Slicer direkt in der jeweiligen Extension ausgeführt werden können.

Die Ordnung dieser Punkte gibt eine grobe Orientierung bezüglich der Reihenfolge während der Umsetzung an. Bei der Bearbeitung der einzelnen Teilaufgaben ist es auch wichtig eine gute Recherche zum aktuellen Stand der Technik durchzuführen. Es ist sehr ungünstig, wenn sich zu Ende eines Projektes herausstellt, dass Lösungen, in die erhebliche Ressourcen investiert wurden, bereits veröffentlicht sind. Um dies zu vermeiden, führt das nächste Kapitel eine umfassende Recherche zu den Teilaufgaben durch.

4.3. Recherche zum Stand der Kunst

Für die Recherche zu den verschiedenen Teilaufgaben ist die Dokumentation der Open Source Plattform 3D Slicer eine wichtige Ressource vgl. 3D Slicer Community 2024a. Diese zeigt bereits etablierte Verfahren und einen *Best Practise* Ansatz. Auch das 3D Slicer Community (2024b) Repository ist eine wichtige Quelle, da so ein Einblick in andere Lösungen möglich ist. So kommt es, das nach ausführlicher Recherche zu den Teilaufgaben UI-Design, Pseudo-Extension, Kapselung Hoffmann, Speicherung

Parameter, Dokumentation und Test bereits Lösungen existieren. Bei diesen Lösungen handelt es sich jedoch nicht um konkrete Ergebnisse, sondern vielmehr um einen Leitfaden zur Lösung der Teilaufgabe. Die Recherche hat demnach ergeben, dass diese Teilprobleme im Kontext der 3D Slicer Umgebung, nicht das erste Mal zutage treten und Lösungswege existieren.

Für ein **UI Design** wird sehr empfohlen, bereits etablierte 3D Slicer Extensions als Orientierung zu nutzen. Eine sehr gute Orientierung bietet das Modul *Transforms*, das in Abbildung 4.2 zu sehen ist. Zu Erkennen ist, dass die UI mittels Collapsible Buttons in verschiedenen Gruppen unterteilt wird. Ohne in die verschiedenen Gruppen hineinzublicken, lässt sich gut abschätzen, welche Parameter wo zu erwarten sind. Dies ermöglicht dem Benutzer ein isoliertes Betrachten der unterschiedlichen Funktionen in diesem Modul und so eine gute Benutzerfreundlichkeit.

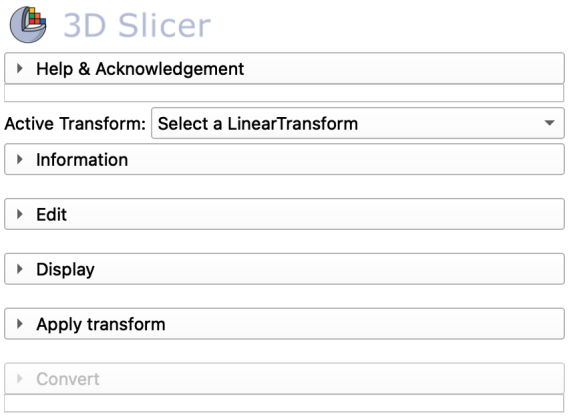

Abbildung 4.2.: *Das Modul Transforms als Beispiel einer etablierten UI für eine Slicer Extension | Screenshot aus 3D Slicer*

Für die Extension, welche in der vorliegenden Arbeit erstellt werden soll, wird genau dieser Ansatz gepflegt und somit eine gute Benutzbarkeit des Moduls gewährleistet. Die speziellen wünsche der konkreten Benutzergruppe sollen jedoch nicht zu kurz kommen.

Für das Erstellen einer **ersten funktionierenden Extension** bietet Slicer eine sehr gute Hilfe. 3D Slicer hat hierfür ein eigenes Modul entwickelt, das sich *Extention Wizard* nennt. Dieses Modul gibt eine gute Einführung in die Entwicklung mit Slicer. Hiermit lässt sich mittels Leitfaden eine erste Demo Extension erstellen, die sich bereits gut in 3D Slicer einfügt. Diese Lösung könnte als Abstraktionsschicht betrachtet werden, da durch dieses Modul im ersten Schritt nahe zu keine Kenntnisse über die Kernanwendung von Slicer nötig sind. Der Extentionwizard ist wie folgt zu finden:

```
Modules -> Developer Tools -> Extension Wizard
```

Für **die Kapselung** einer bestimmten Einheit von Code, sieht Slicer eine Bibliothek innerhalb der Extention vor, so beschreibt es die 3D Slicer Community (2024a). Das Listing 4.1 zeigt, wo eine Bibliothek innerhalb einer Extention einzuordnen ist, hier als `MyLib` zu sehen. Innerhalb dieses Ordners können sich weitere Module befinden, die als einfache Python-Files abgelegt werden. Zu beachten ist, dass in jeder Bibliothek

eine `__init__.py` hinzugefügt wird, sodass dieser Ordner auch entsprechend verwendet werden kann.

```
|-- CMakeLists.txt
|-- MyLib
|   |-- __init__.py
|   |-- cool_maths.py
|   |-- utils.py
|-- MyExtension.py
```

4.1.Listing.: *Prinzipeller Aufbau eines Projektes für eine Slicer Extension nach Slicer (2024)*

Für die Teilaufgabe zur **Speicherung der Parameter** nutzt Slicer eine Technik, das durchaus als eines der Kernfunktionen beschrieben werden kann. Die Rede ist hier von der Klasse `ParameterNodeWrapper`. Dieser wurde bereits zu einem früheren Zeitpunkt in dieser Arbeit beschrieben. Für die Funktion dieser Lösung sei auf den Abschnitt 2.5.4 Benutzerschnittstelle verwiesen.

Sowohl das **Benutzerhandbuch**, also auch die technische Dokumentation einer Slicer Extension wird immer in der `README.md` des jeweiligen Repository hinterlegt. In der Extension selber sieht Slicer keine umfangreiche Dokumentation vor. Es wird lediglich via Link auf die Dokumentation im Repository verwiesen und eine kurze Einführung gegeben. Auch hier gibt es wieder etablierte Designs, die als Vorlage verwenden werden können. Die 3D Slicer Community (2024a) erwähnt hier unter anderem das Module SegmentMesher als gutes Beispiel.

Für die Letzte Teilaufgabe, das **Testen**, gibt die Dokumentation von 3D Slicer ebenfalls eine konkrete Struktur vor. Ist ist also nicht nötig eigenen Verfahren zu entwickeln. Für die Softwaretests stell Slicer eine eigene Klasse innerhalb der Slicer Bibliothek bereit, die alle wichtigen Funktionen enthält, um konkrete Testfälle zu schreiben. Die Klasse Trägt den Namen *ScriptedLoadableModuleTest* und kann als Elternklasse für einen Testfall verwendet werden.

```
class MyExtensionTest(ScriptedLoadableModuleTest)
    def setUp(self):
      # do something befor a test
    def runTest(self):
      # execute your test case here
    def testMyExtension1(self):
      # write your test case here
```

4.2.Listing.: *Aufbau einer Testklasse zum ausführen von Unittest nach 3D Slicer Community (2024a)*

Im Listing 4.2 ist zu sehen, dass sich die Klasse in drei Teil aufteilt, die Methode `setUp()`, die als Vorbedingung dient, die Methode `runTest()`, die über die UI getriggert werden kann um die Test zu starten und die konkreten Textfälle, die in selbst definierten Methoden geschrieben weden können. Als Beispiel sie hier die Methode `testMyExtension1()` gezeigt.

Im Laufe dieses Kapitels wurde klar, dass es für einige der Teilaufgaben bereits Lösungen oder Lösungsansätze gibt, die zu etwas weniger Entwicklungsarbeit führen. Jeodch trifft dies nicht auf alle Punkte zu, sodass sich der nächste Abschnitt mit der Erarbeitung von konkreten Lösungsansäätzen für die noch nicht behandelten Teilaufgaben beschäftigt.

4.4. Erarbeiten von Lösungsansätzen

hier geht es um Brainstorming

Architekturdesign

Single Prozess

Batch Prozess

4.5. Auswahl von Lösungsansätzen

5. Ergebnisse

5.1. Metainormationen

Hier steht eine allgemine Beschreibung der Extension Name, ersteller, ...

5.2. Konzeptionen

Diagramme

5.3. Implementierungen

Wichtige Codeausschnitte

6. Diskussion und Limitierungen

test

7. Schlussfolgerung und Ausblick

test

Erklärung zur Abschlussarbeit

Hiermit versichere ich, die eingereichte Abschlussarbeit selbständig verfasst und keine andere als die von mir angegebenen Quellen und Hilfsmittel benutzt zu haben. Wörtlich oder inhaltlich verwendete Quellen wurden entsprechend den anerkannten Regeln wissenschaftlichen Arbeitens zitiert. Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht anderweitig als Abschlussarbeit eingereicht wurde. Das Merkblatt zum Täuschungsverbot im Prüfungsverfahren der Hochschule Augsburg habe ich gelesen und zur Kenntnis genommen. Ich versichere, dass die von mir abgegebene Arbeit keinerlei Plagiate, Texte oder Bilder umfasst, die durch von mir beauftragte Dritte erstellt wurden.

Augsburg, den 30. Dezember 2024

_____

Lukas Konietzka

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

A. Anhang

test

Literaturverzeichnis

3D Slicer Community (2024a). *3D Slicer: A multi-platform, free and open source software package for visualization and image analysis*. https://www.slicer.org. Zugriff am 21. November 2024 (siehe S. 17–22, 27–29, 35, 37).

3D Slicer Community (2024b). *Slicer Extensions Index*. https://github.com/Slicer/ExtensionsIndex. GitHub repository (siehe S. 18, 27, 35).

Baird, Emily und Gavin Taylor (2017). „X-ray micro computed-tomography“. In: *Current Biology* 27.8, R289–R291 (siehe S. 7).

Burger, Wilhelm und Mark James Burge (2009). *Digitale Bildverarbeitung: Eine Algorithmische Einführung Mit Java*. Springer-Verlag (siehe S. 9, 14).

Buzug, Thorsten M (2011). „Computed tomography“. In: *Springer handbook of medical technology*. Springer-Verlag, S. 311–342 (siehe S. 7).

Crespigny, Alex de, Hani Bou-Reslan, Merry C Nishimura, Heidi Phillips, Richard AD Carano und Helen E D'Arceuil (2008). „3D micro-CT imaging of the postmortem brain“. In: *Journal of neuroscience methods* 171.2, S. 207–213 (siehe S. 3).

Diwakar, Manoj und Manoj Kumar (2018). „A review on CT image noise and its denoising“. In: *Biomedical Signal Processing and Control* 42, S. 73–88 (siehe S. 10).

Fedorov, Andrey, Reinhard Beichel, Jayashree Kalpathy-Cramer, Julien Finet, Jean-Christophe Fillion-Robin, Sonia Pujol, Christian Bauer, Dylan Jennings, Fiona M. Fennessy, Milan Sonka, John Buatti, Stephen R. Aylward, James V. Miller, Steve Pieper und Ron Kikinis (Nov. 2012). „3D Slicer as an Image Computing Platform for the Quantitative Imaging Network“. In: *Magnetic Resonance Imaging* 30.9, S. 1323–1341. DOI: 10.1016/j.mri.2012.05.001 (siehe S. 17–20, 35).

Handels, Heinz (2000). *Medizinische Bildverarbeitung*. Springer-Verlag (siehe S. 2, 10–13, 35).

Hoffmann, Simon (Jan. 2020). „Unterstützung der Karies-Klassifizierung in Mikro-CT-Aufnahmen durch 3D-Bildverarbeitung“. Bachelorarbeit. Technische Hochschule Augsburg (siehe S. 5, 14, 16, 24, 35).

ITK Development Team (2024). *ITK: Insight Segmentation and Registration Toolkit*. https://itk.org. Entwickelt und gepflegt von Kitware, Zugriff am 21. November 2024 (siehe S. 21).

Jähne, Bernd (2024). *Digitale bildverarbeitung: und bildgewinnung*. Springer-Verlag (siehe S. 9).

Lehmann, Klaus M, Elmar Hellwig und Hans-Jürgen Wenz (2012). *Zahnärztliche Propädeutik: Einführung in die Zahnheilkunde; mit 32 Tabellen*. Deutscher Ärzteverlag (siehe S. 5, 6, 35).

Lehmann, Thomas, Walter Oberschelp, Erich Pelikan und Rudolf Repges (2013). *Bildverarbeitung für die Medizin: Grundlagen, Modelle, Methoden, Anwendungen*. Springer-Verlag (siehe S. 7, 12–14, 35).

National Institute of Biomedical Imaging and Bioengineering (NIBIB) (2024). *X-Rays - Science Topic*. Zugriff am: 2024-11-15. URL: https://www.nibib.nih.gov/science-education/science-topics/x-rays (siehe S. 7, 8).

Poliklinik (2024). *Poliklinik für Zahnerhaltung und Parodontologie des LMU-Klinikums München*. https://www.klinikum.uni-muenchen.de/Poliklinik-fuer-Zahnerhaltung-und-Parodontologie/. Zugriff am 22. November 2024 (siehe S. 2, 9, 35).

Qt Development Team (2024). *Qt Documentation: Introduction to Qt*. https://doc.qt.io/qt-6/qt-intro.html. Entwickelt und gepflegt von The Qt Company, Zugriff am 21. November 2024 (siehe S. 22).

Ramesh, KKD, G Kiran Kumar, K Swapna, Debabrata Datta und S Suman Rajest (2021). „A review of medical image segmentation algorithms“. In: *EAI Endorsed Transactions on Pervasive Health and Technology* 7.27, e6–e6 (siehe S. 12).

Rösch, Peter und Karl-Heinz Kunzelmann (2018). „Efficient 3D Rigid Registration of Large Micro CT Images“. In: *International Journal of Computer Assisted Radiology and Surgery;* Bd. 13. June 2018, issue 1 (supplement), S. 118–119. DOI: 10.1007/s11548-018-1766-y (siehe S. 8).

SCANCO Medical AG (2024). *Brochures and Documentation*. Zugriff am: 22. November 2024 (siehe S. 8).

VTK Development Team (2024). *VTK: The Visualization Toolkit*. https://vtk.org. Entwickelt und gepflegt von Kitware, Zugriff am 21. November 2024 (siehe S. 21).

Zwinkels, Joanne (2015). „Light, electromagnetic spectrum“. In: *Encyclopedia of Color Science and Technology* 8071, S. 1–8 (siehe S. 7, 35).