

Hochschule Luzern - Technik & Architektur

3D-Segmentierer basierend auf freier Software

Bachelor-Thesis Medizintechnik, FS19

Adrian von Wyl

Industriepartner

Sven Kunkel

Hochschule Mannheim

Betreuer

Prof. Dr. Philipp Schütz

Experte

Dr. Iwan Jerjen

Bachelor-Thesis an der Hochschule Luzern - Technik & Architektur

Titel 3D-Segmentierer basierend auf freier Software

Diplomandin/Diplomand von Wyl, Adrian

Bachelor-Studiengang Bachelor Medizintechnik

Semester FS19

Dozentin/Dozent Schütz, Philipp

Expertin/Experte Jerjen, Iwan

Abstract Deutsch

Die vorliegende Arbeit befasst sich mit dem Prozess der Implementierung eines Segmentierungs-Werkzeuges, welches auf freier Software basiert. Um Daten aus der Computer Tomographie (CT) für die Entwicklung von Implantaten zu nutzen, müssen aus den 3D-Bildstapeln erst die Oberflächen der einzelnen Organe oder knöchernen Strukturen extrahiert werden. Hierfür stehen bereits einige offene Werkzeuge zur Verfügung, die jedoch qualitativ nicht auf dem Stand von Industrie-Software sind. Die Industrie-Werkzeuge sind aufgrund ihrer Kosten aber für kleine Entwicklungsprojekte nicht finanzierbar. Daher besteht ein Bedarf an einer offenen, qualitativ hochstehenden Segmentierlösung. Ausgehend von den Ergebnissen der durchgeführten Literaturrecherche und einer Analyse von unterschiedlichen Segmentierungsprogrammen wurden zwei Prototypen implementiert, welche unterschiedliche Ansätze in der Segmentierung verfolgen. Beide Prototypen eignen sich zum Segmentieren von Bilddatensätzen aus dem CT, wobei beide Prototypen in ihren Eigenschaften Vor- und Nachteile aufweisen.

Abstract Englisch

This paper deals with the process of implementing a segmentation tool based on free software. In order to use data from computer tomography (CT) for the development of implants, the surfaces of the individual organs or bony structures must first be extracted from the 3D image stacks. Some open tools are already available for this purpose, but their quality is not up to date with industrial software. However, industrial tools are not affordable for small development projects due to their cost. Therefore, there is a need for an open, high-quality segmentation solution. Based on the results of the literature research carried out and an analysis of different segmentation programs, two prototypes were implemented, which pursue different approaches in segmentation. Both prototypes are suitable for segmenting image data sets from CT, whereby both prototypes have advantages and disadvantages in their properties.

Ort, Datum Horw, 07.06.2019

© Adrian von Wyl, Hochschule Luzern – Technik & Architektur

Alle Rechte vorbehalten. Die Arbeit oder Teile davon dürfen ohne schriftliche Genehmigung der Rechteinhaber weder in irgendeiner Form reproduziert noch elektronisch gespeichert, verarbeitet, vervielfältigt oder verbreitet werden.

Sofern die Arbeit auf der Website der Hochschule Luzern online veröffentlicht wird, können abweichende Nutzungsbedingungen unter Creative-Commons-Lizenzen gelten. Massgebend ist in diesem Fall die auf der Website angezeigte Creative-Commons-Lizenz.

Inhaltsverzeichnis

1	EINLEITUNG	1
1.1	AUSGANGSSITUATION	1
1.1.1	Pixelorientierte Verfahren	1
1.1.2	Kantenbasierte Verfahren	3
1.1.3	Regionenorientierte Verfahren	3
1.1.4	Modellbasierte Verfahren	4
1.1.5	Hybride Segmentierungsverfahren	5
1.2	PROBLEMSTELLUNG	5
1.3	ZIELSETZUNG	5
1.4	STRUKTUR DER ARBEIT	5
2	MATERIAL UND METHODEN	6
2.1	ANALYSE VON SEGMENTIERUNGS-PROGRAMMEN	6
2.1.1	3D Slicer	6
2.1.2	ITK-SNAP	8
2.1.3	VGSTUDIO MAX	11
2.2	IMPLEMENTIERUNG EINES SEGMENTIERUNGSPROTOTYPS	13
2.2.1	Prototyp «Alpha»	13
2.2.2	Prototyp «Beta»	14
2.3	ÜBERPRÜFUNG/VALIDIERUNG DES SEGMENTIERUNGSPROTOTYPS	17
2.3.1	Voraussetzungen	17
2.3.2	Durchführung	18
3	RESULTATE	19
3.1	ANALYSE DER SEGMENTIERUNG-PROGRAMME	19
3.1.1	3D Slicer	19
3.1.2	ITK-SNAP	19
3.1.3	VGSTUDIO MAX	20
3.1.4	Fazit	21
3.2	RESULTATE DES SMOOTHPOLYDATAFILTER-ALGORITHMUS	21
3.3	VALIDIERUNG	23
3.3.1	Testfall 1 – «Thorax»	23
3.3.2	Testfall 2 – «Truncus»	26
3.3.3	Testfall 3 – «Cranium»	29
3.3.4	Fazit	32
4	SCHLUSSFOLGERUNG	33
4.1	ZUSAMMENFASSUNG DER ERGEBNISSE	33
4.2	ZUKÜNFTIGER FORSCHUNGSBEDARF	34
	DANKSAGUNG	35
	LITERATURVERZEICHNIS	36
	ANHANG	37

1 Einleitung

1.1 Ausgangssituation

Die Segmentierung ist ein wichtiges Teilgebiet in der medizinischen Bildverarbeitung. Sie dient als Grundlage für die weitergehende Analyse, Vermessung und 3D-Visualisierung der Daten. Als Segmentierung kann allgemein die hervorgehobene Darstellung von inhaltlich zusammenhängenden Regionen durch Zusammenfassung benachbarter Bildpunkte (Pixel) oder Volumenelemente (Voxel) definiert werden. Die zusammenhängenden Regionen genügen dabei einem bestimmten Homogenitätskriterium (Lehmann, Oberschelp, Pelikan, & Reppes, 1997). Im Laufe der Jahre wurden verschiedene Verfahren zur manuellen Segmentation entwickelt. Diese lassen sich grob in folgende Kategorien einteilen:

- Pixelorientierte Verfahren
- Kantenbasierte Verfahren
- Regionenorientierte Verfahren
- Modellbasierte Verfahren

Auch eine Kombination der oben genannten Verfahren ist möglich. Diese Verfahren werden dann als hybride Segmentierungsverfahren zusammengefasst.

In diesem Kapitel sollen nun die wichtigsten Segmentierungsverfahren genauer beschreiben werden.

1.1.1 Pixelorientierte Verfahren

Bei diesem Verfahren wird über ein bestimmtes Attribut bei jedem einzelnen Pixel über die Zugehörigkeit zur jeweiligen Region entschieden. Dabei wird nur das momentane Pixel berücksichtigt, ohne dabei dessen Umgebung zu analysieren. Dadurch kann bei diesem Verfahren nicht gewährleistet werden, dass tatsächlich immer nur zusammenhängende Segmente entstehen. Eine Nachbearbeitung ist deshalb oft notwendig, z.B. durch morphologische Filterung.

Die meisten pixelorientierten Verfahren basieren auf Schwellenwertverfahren. Dabei entscheidet allein der Grauwert des Pixels darüber, zu welcher Region es zugeteilt wird. Überschreitet der Grauwert eine zuvor festgelegte Schwelle, gehört es zur entsprechenden Region. Ist die Zuordnung der Pixelhelligkeit zum Gewebetyp konstant und bekannt, können **statische Schwellenwerte** verwendet werden. Diese sind unabhängig von der jeweiligen Aufnahme. In Abbildung 1 ist eine statische Schwellenwertsegmentierung eines CT-Schnittbildes zu sehen. Dabei werden Knochen- oder Weichteilfenster mit statischen Schwellenwerten auf den Hounsfield-Einheiten (HU) realisiert. Die Hounsfield-Einheiten werden auf der Hounsfield-Skala angegeben. Dabei handelt es sich um eine quantitative Skala, welche die Abschwächung von Röntgenstrahlung im Gewebe beschreibt und in Graustufenbildern darstellt. Die Werte können so Gewebearten zugeordnet werden. Die Tabelle 1 zeigt typische Hounsfield-Werte für unterschiedliche Gewebearten.

Bei **dynamischen Schwellenwerten** wird das Bild zunächst analysiert. Anhand dieser Analyse wird der Schwellenwert individuell für dieses Bild bestimmt. Werden innerhalb eines Bildes an verschiedenen Orten unterschiedliche Schwellenwerte ermittelt, spricht man von **adaptiven Schwellenwerten**. Dies führt im Extremfall dazu, dass für alle Pixelpositionen ein eigener Schwellenwert ermittelt wird. Das kann notwendig sein, wenn aufgrund kontinuierlicher Helligkeitsverläufe im Bild das dynamische Schwellenwertverfahren nicht ausreicht (Deserno, 2011).

Pixelorientierte Verfahren sind schnell und führen bei homogenen Intensitätsgebieten zu guten Ergebnissen. Wie bereits angesprochen, besteht die Gefahr, dass nicht zusammenhängende Regionen entstehen können. Daher eignet sich dieses Segmentierungsverfahren beispielsweise für die Unterscheidung von Knochen, Weichteilen und Hintergrund in CT-Bildern.

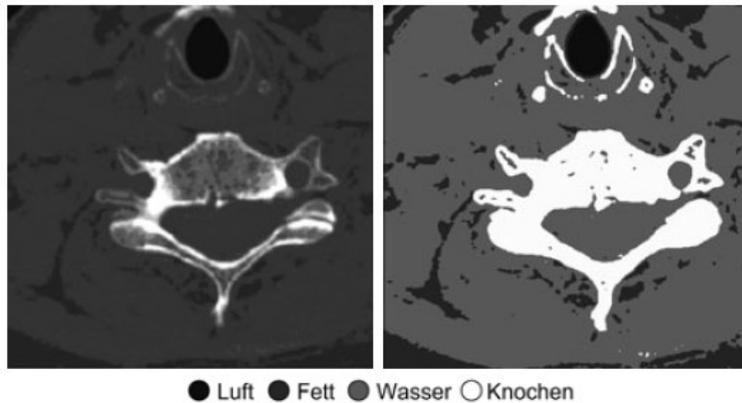


Abbildung 1: Statische Schwellenwertsegmentierung eines CT-Schnittbildes. Der CT-Schnitt aus dem Bereich der Wirbelsäule kann statisch segmentiert werden, da durch die Normierung der Hounsfield-Einheiten (HU) auf einen Bereich $[-1000, 3000]$ sog. Knochen- $[200, 3000]$ oder Weichteilfenster für Wasser $[-200, 200]$, Fett und Gewebe $[-500, -200]$ sowie Luft $[-1000, -500]$ fest definiert werden können (Deserno, 2011).

Tabelle 1: Typische Hounsfield-Werte für unterschiedliche Gewebearten (Hebb & Poliakov, 2009)

Substance	HU
Air	-1000
Lung	-500
Fat	-100 to -50
Water	0
CSF	15
Kidney	30
Blood	+30 to +45
Muscle	+10 to +40
Grey matter	+37 to +45
White matter	+20 to +30
Liver	+40 to +60
Soft Tissue, Contrast	+100 to +300
Bone	+700 (cancellous bone) to +3000 (dense bone)

1.1.2 Kantenbasierte Verfahren

Bei kantenbasierten Verfahren wird versucht, die im Bild enthaltenen Objekte aufgrund ihrer Kanten zu erfassen. Daher eignen sich solche Verfahren nur für jene Fälle, in welchen die Objekte mit klar definierten Umrandungen dargestellt werden. Dies ist jedoch in der medizinischen Bildverarbeitung nur in Ausnahmen der Fall, so zum Beispiel bei der Aufnahme metallischer Implantate, welche sich auf Röntgenbildern klar vom Hintergrund abgrenzen.

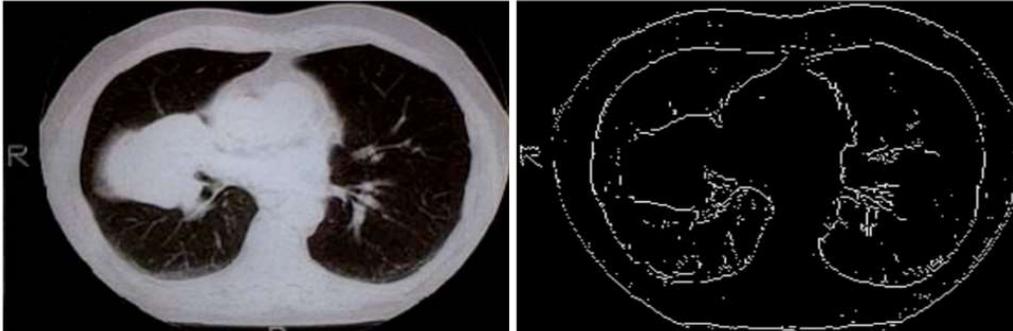


Abbildung 2: Segmentierung der Lunge (links) durch Anwendung eines kantenbasierten Segmentierungsverfahrens (rechts) (Dua, Kandiraju, & Chowirappa, 2009).

Bei Objektgrenzen erreicht der Helligkeitsgradient des Bildes ein betragsmässiges Maximum. Diese Eigenschaft macht sich das kantenbasierte Verfahren zu Nutze, um die Kante eines Objekts zu finden und diese mit einem entsprechenden Algorithmus zu verfolgen. Ist die Kante nicht vollständig vorhanden, kann auch keine Segmentierung erfolgen, was einen wesentlichen Nachteil dieser Methode darstellt.

1.1.3 Regionenorientierte Verfahren

Regionenorientierte Verfahren sind iterative Verfahren, bei welchen die Merkmalsberechnung und Segmentierung abwechselnd aufeinander folgen (Jähne, 2012). Dazu wird in einem ersten Schritt der Startpunkt (Seed) spezifiziert. Anschliessend werden die Nachbarelemente betrachtet und auf ein für die Region spezifisches Attribut untersucht. Aufgrund dessen werden die untersuchten Elemente in die jeweilige Region eingeteilt. Anschliessend werden weitere Nachbarelemente untersucht. Dieser Prozess wiederholt sich so lange, bis sich keine neuen Elemente mehr entsprechend des gewählten Attributs einteilen lassen.

Auf solchen Verfahren basieren Algorithmen wie das Region Growing, das Split-and-Merge Verfahren oder auch das Pyramid Linking. In Abbildung 3 ist die Segmentierung der Hüftknochen durch Verwendung des Region Growing Algorithmus zu sehen.

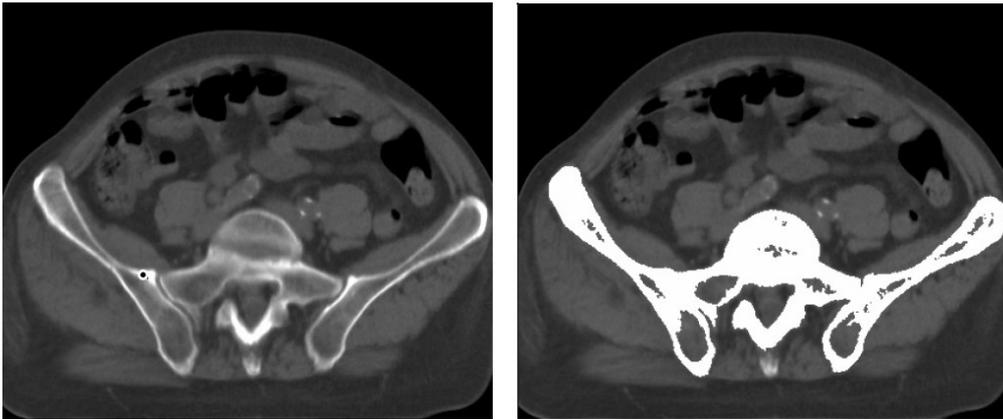


Abbildung 3: Ausgehend von dem Startpunkt (links) wird der äussere Teil der Hüftknochen segmentiert, die sich durch hohe Hounsfield-Werte auszeichnen (rechts) (Handels, 2009).

1.1.4 Modellbasierte Verfahren

Wie bereits angesprochen, kann mit kantenbasierten Verfahren keine Segmentierung erfolgen, wenn die Kanten nicht vollständig vorhanden sind. Dies ist auch bei regionenorientierten Verfahren der Fall. Aus diesem Grund wurden modellbasierte Verfahren entwickelt, welche sich an der menschlichen visuellen Fähigkeit orientieren, Formen zu erkennen, selbst wenn diese nicht vollständig dargestellt sind. Dies ist allerdings nur möglich, wenn die Form vorher schon bekannt ist. Einer dieser Effekte ist in Abbildung 4 zu sehen.

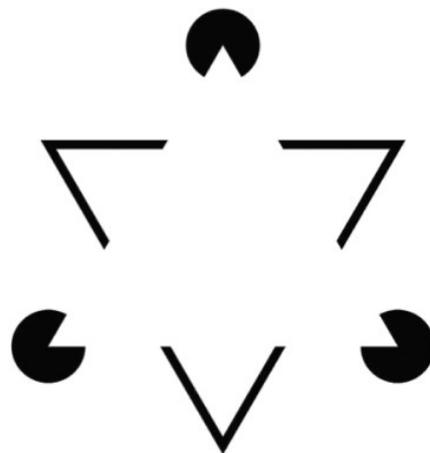


Abbildung 4: Dreieck Illusion (Tarjetas, o.D.)

Genau nach diesem Prinzip sind solche modellbasierten Verfahren aufgebaut. Sie werden eingesetzt, wenn die exakte Form der im Bild enthaltenen Objekte bekannt ist. Möchte man beispielsweise eine Segmentierung eines Dreiecks durchführen, so wird der Datensatz genau nach dieser Form durchsucht. Auf diese Weise lassen sich auch andere Formen segmentieren. Ein Beispiel für ein modellbasiertes Verfahren ist die Houghstransformation (Jähne, 2012).

1.1.5 Hybride Segmentierungsverfahren

Da jede der aufgeführten Verfahren in der Regel Vor- und Nachteile hat, wurden hybride Segmentierungsverfahren entwickelt. Diese Verfahren kommen in der Praxis der medizinischen Bildverarbeitung die grösste Bedeutung zu. Das Ziel solcher Mischverfahren ist es, die Vorteile einzelner (meist kanten- und regionenorientierter) Algorithmen miteinander zu verbinden, um so die Nachteile der gewählten Verfahren auszugleichen (Deserno, 2011). Weit verbreitete hybride Segmentierungsverfahren sind die Wasserscheiden-Transformation und die aktiven Konturmodelle.

1.2 Problemstellung

Um CT-Daten für die Entwicklung von Implantaten zu nutzen, müssen aus den 3D-Bildstapeln erst die Oberflächen der einzelnen Organe oder knöchernen Strukturen extrahiert werden. Hierfür stehen bereits einige offene Werkzeuge zur Verfügung, die jedoch qualitativ nicht mit Industrie-Software mithalten können. Die Industrie-Werkzeuge sind aufgrund ihrer Kosten aber für kleine Entwicklungsprojekte nicht finanzierbar. Daher besteht ein Bedarf an einer offenen, qualitativ hochstehenden Segmentierlösung.

1.3 Zielsetzung

In diesem Projekt soll auf Basis offener Software ein Segmentierungs-Werkzeug implementiert werden, das die qualitative Lücke zwischen offenen Produkten und Industrie-Werkzeugen zu schliessen beginnt. Das aus der Arbeit resultierende Werkzeug soll dann in kleineren Forschungsarbeiten eingesetzt werden können.

1.4 Struktur der Arbeit

In der Einleitung wurden bereits einige wichtige Grundlagen vermittelt, die für das Verständnis der Arbeit notwendig sind. Dabei standen die unterschiedlichen Ansätze, wie man eine Segmentierung durchführt, im Vordergrund. Ausserdem wurde auf die Problemstellung und das Ziel der Arbeit eingegangen.

Im folgenden Kapitel wird der Prozess beschreiben, der während des Projekts durchlaufen wurde, und welche Mittel dazu eingesetzt wurden. Dabei werden zunächst unterschiedliche, bereits bestehende Segmentierungs-Programme vorgestellt und analysiert. Anschliessend wird die Implementierung von zwei Prototypen erläutert sowie das Vorgehen bei der Überprüfung und Validierung der Prototypen erklärt.

Im Kapitel 3 werden die Ergebnisse der Analyse der Segmentierungs-Programme vorgestellt sowie die Resultate der Überprüfung und Validierung der implementierten Prototypen zusammengefasst und mit den bestehenden Programmen verglichen.

Abschliessend wird in Kapitel 4 ein Fazit gezogen und mögliche Verbesserungen der implementierten Prototypen angesprochen.

2 Material und Methoden

In diesem Kapitel werden die untersuchten Segmentierungsapplikationen präsentiert und deren Untersuchungsmethode vorgestellt. Des Weiteren wird aufgezeigt, wie man bei der Implementierung des eigenen Segmentierungsprototyps vorgegangen ist.

2.1 Analyse von Segmentierungs-Programmen

Für die Analyse der Applikationen wurde beispielhaft die Segmentierung der rechten Niere durchgeführt und die generierte Oberfläche als STL-Datei exportiert. Dazu wurde jeweils derselbe Bilddatensatz verwendet, um die Ergebnisse der Analyse der unterschiedlichen Programme vergleichbar zu machen. Der Bilddatensatz, welcher mittels Computertomographie (CT) generiert wurde, zeigt den Oberkörper eines männlichen Patienten. Der Datensatz stammt aus dem Data Store der freien Software 3D Slicer.

2.1.1 3D Slicer

3D Slicer ist eine Open-Source-Softwareplattform für medizinische Bildinformatik, Bildverarbeitung und dreidimensionale Visualisierung. Die Entwicklung dieser Software begann 1998 im Rahmen eines Masterarbeitsprojekts zwischen dem Surgical Planning Laboratory des Brigham and Women's Hospital in Boston und dem MIT Artificial Intelligence Laboratory in Cambridge. Seither wurde 3D Slicer mit Unterstützung der National Institutes of Health und einer weltweiten Entwicklergemeinschaft weiterentwickelt und bietet Ärzten, Forschern und der Öffentlichkeit kostenlose, plattformübergreifende Verarbeitungswerkzeuge. Die erste komplett neu überarbeitete Version wurde 2007 veröffentlicht (3D Slicer, o.D.). Im Rahmen dieser Arbeit wurde die Version 4.8.1 verwendet.

Vorgehen

Der Bilddatensatz wurde aus dem Data Store von 3D Slicer heruntergeladen und eingelesen. Dabei handelt es sich um einen Datensatz im DICOM-Format. Die Abbildung 5 zeigt das User Interface von 3D Slicer. Die Daten werden auf allen drei Schnittebenen (Sagittal-, Transversal- und Frontalebene) dargestellt.

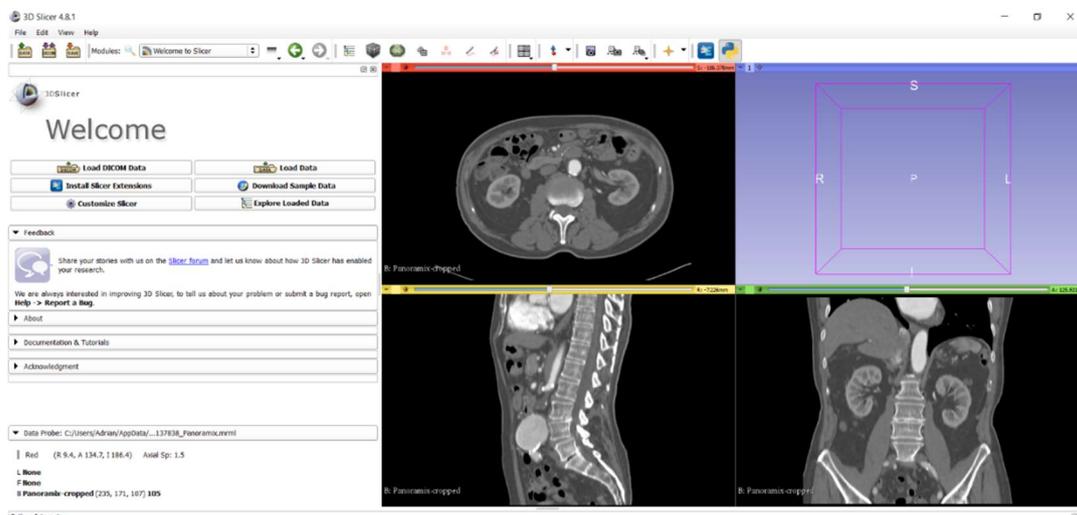


Abbildung 5: 3D Slicer User Interface

Anschliessend lässt sich die Region of Interest (ROI) bestimmen. Diese wird so gewählt, dass nur das gewünschte Gewebe, in diesem Fall die Niere, angezeigt wird. Jenes Gewebe, welches nicht innerhalb der ROI liegt, wird ausgeblendet.

Über den Editor lassen sich verschiedene Algorithmen anwenden. In diesem Beispiel eignet sich der Algorithmus ThresholdEffect, welcher dem Schwellenwertverfahren, also einem pixelorientierten Segmentierungsverfahren, entspricht. Der Schwellenwertbereich wurde auf 50 – 377 festgelegt, wobei 377 dem Maximum entspricht. Sämtliche Voxel, welche sich in diesem Bereich befinden, werden so in ein vorher bestimmtes Label eingeteilt (Abbildung 6). 3D Slicer stellt dazu eine grosse Anzahl an unterschiedlichen Labels zur Verfügung. Für diese Segmentierung wurde passenderweise das Label «right kidney» gewählt.

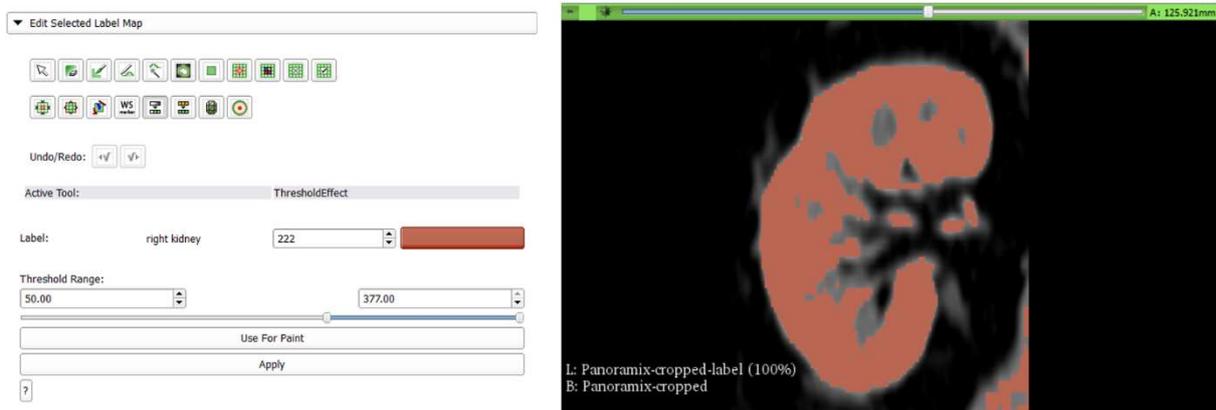


Abbildung 6: Schaltfläche und Ergebnis der Schwellenwertsetzung

Ungewünschtes Gewebe wurde anschliessend durch eine Kombination von Erosions- und Dilationsalgorithmen entfernt (Abbildung 7 und 8).

Über die Schaltfläche Split Merge Volume wird das segmentierte Objekt in die Liste übernommen und 3-dimensional visualisiert. Das entstandene Modell wurde schliesslich als STL-Datei exportiert und gespeichert. Die 3D-Visualisierung sowie der STL-Export wird in Kapitel Resultat genauer vorgestellt.

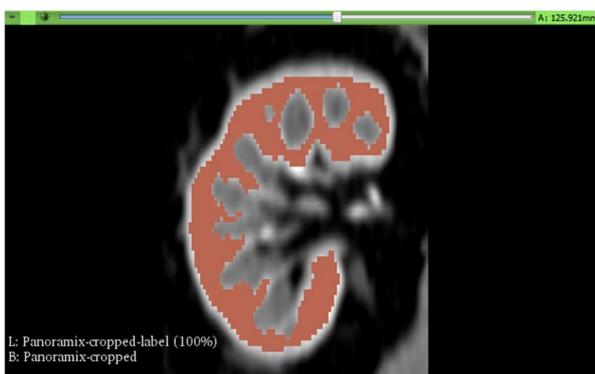


Abbildung 7: Ergebnis nach Erosion



Abbildung 8: Ergebnis nach Dilatation

2.1.2 ITK-SNAP

ITK-SNAP ist eine interaktive Softwareanwendung, mit der Benutzer dreidimensionale medizinische Bilder navigieren, anatomische Interessensgebiete manuell abgrenzen und eine automatische Bildsegmentierung durchführen können. Die Software wurde über Jahrzehnte in Zusammenarbeit mit zwischen Paul Yushkevich, Ph.D., des Penn Image Computing and Science Laboratory (PICS) an der University of Pennsylvania, und Guido Gerig, Ph.D., des Scientific Computing and Imaging Institute (SCI) an der University of Utah, entwickelt. ITK-SNAP ist kostenfrei, Open-Source und plattformübergreifend (ITK-SNAP, 2018). In der vorliegenden Arbeit wurde mit der aktuellsten Version ITK-SNAP 3.8.0-beta gearbeitet. Diese Version wurde am 28. Oktober 2018 veröffentlicht.

Vorgehen

Als Bilddatensatz wurde der bereits beim 3D Slicer verwendete Datensatz verwendet, um die Ergebnisse vergleichbar zu machen. Der Datensatz wurde zunächst importiert und anschliessend in den drei bereits bekannten Körperebenen dargestellt (Abbildung 9). Ebenfalls in dieser Abbildung ersichtlich ist die bereits manuell festgelegte ROI.

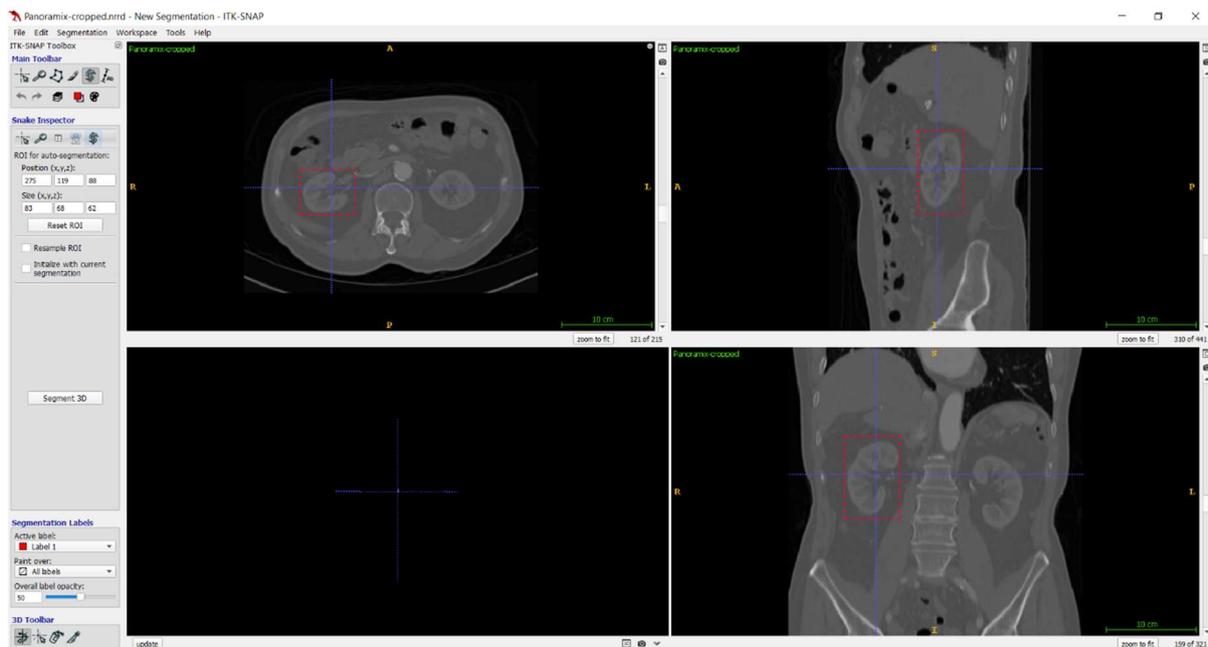


Abbildung 9: ITK-SNAP User Interface mit bereits definiertem ROI

Nach der Bestimmung des Labels gelangt man über die Schaltfläche Segment 3D ins Segmentierungsmenü. Dort wird zunächst eine Vorsegmentierung durchgeführt. Diese basiert ebenfalls auf einem Schwellenwertverfahren. Der Schwellenwert wird, wie bereits beim 3D Slicer, auf 50 – 371 gesetzt, wobei hier bei 371 bereits das Maximum erreicht ist. Beim Einstellen des Schwellenwerts fällt auf, dass nicht die Niere selbst, sondern dessen Umgebung markiert wird (Abbildung 10).

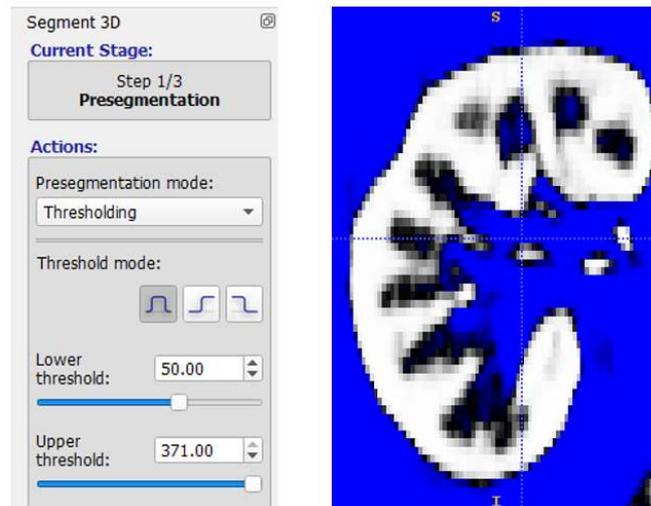


Abbildung 10: Schaltfläche und Ergebnis der Schwellenwertsetzung

In einem nächsten Schritt wird ein Region Growing durchgeführt. Dabei handelt es sich um ein regionenorientiertes Segmentierungsverfahren, bei welchem zunächst ein Startpunkt (Seed) in Form einer Kugel (hier: «Bubble») in die Schwellenwertregion gesetzt wird. Der Durchmesser der Kugel kann entsprechend der Strukturgröße des zu segmentierenden Objekts angepasst werden (Abbildung 11). Die einzelnen Schritte des Region Growings können vom Benutzer sowohl manuell als auch automatisch über die Play-Schaltfläche durchgeführt werden. Dabei wird die Anzahl der Iterationen direkt angezeigt (Abbildung 12). Einzelne Iterationen lassen sich zudem rückgängig machen. Sind alle Iterationen durchlaufen (Abbildung 13), lässt sich durch Wählen der Schaltfläche Update Mesh das segmentierte Objekt 3-dimensional darstellen. Die erhaltende Oberfläche lässt sich auch hier als STL-Datei exportieren und speichern.

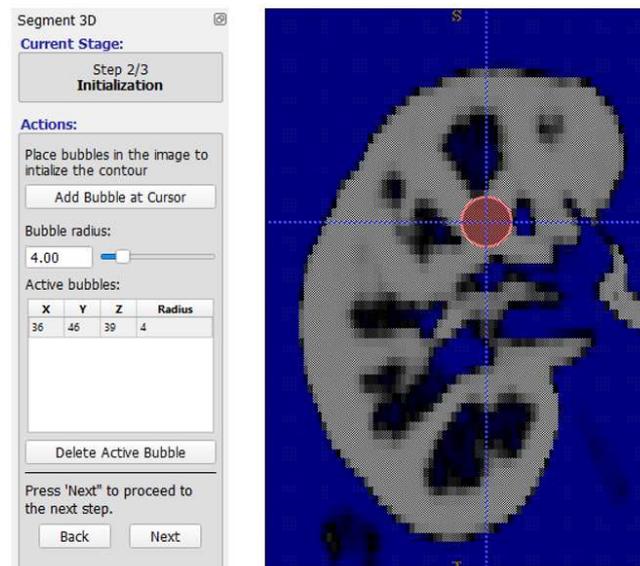


Abbildung 11: Segmentierungsfenster und Setzen eines „Bubbles“ auf einer der Ebenen

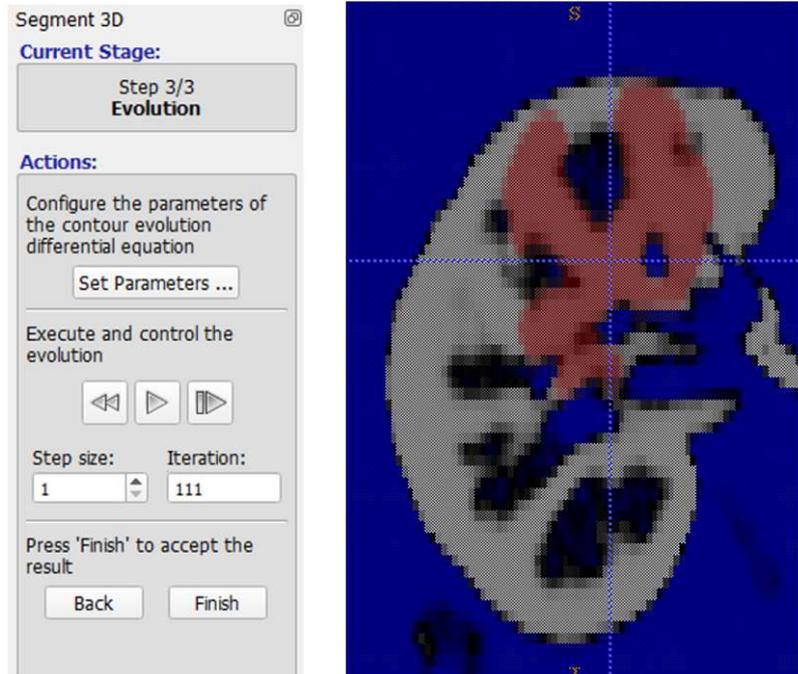


Abbildung 12: Segmentierungsfenster und Zwischenschritt des Region Growings (Iteration 111)

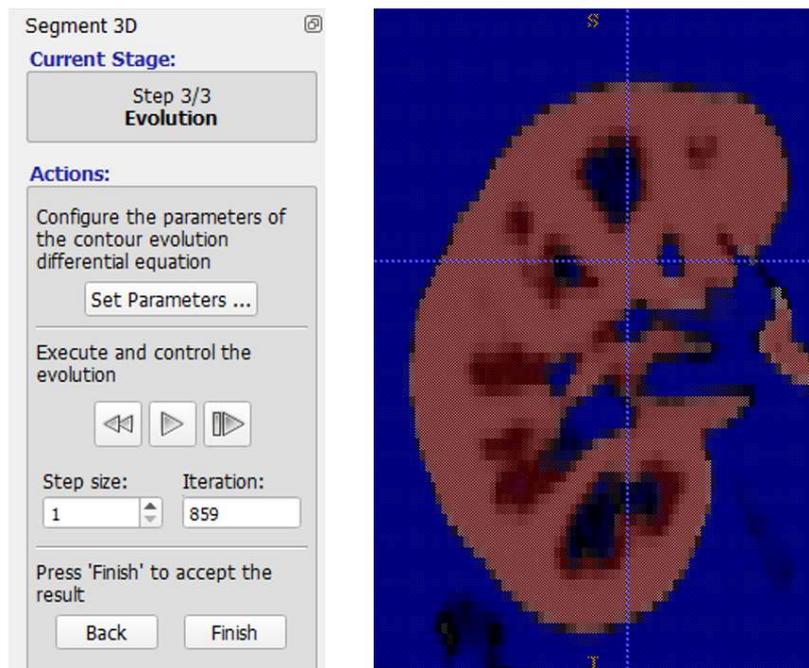


Abbildung 13: Segmentierungsfenster und Ergebnis des Region Growings

2.1.3 VGSTUDIO MAX

VGSTUDIO ist eine Software für die Verarbeitung von Voxeln. Sie war das erste Computer-Programm zur Analyse und Volumenvisualisierung von Computertomographie- sowie Magnetresonanztomographie-Daten (MRT) für die Zwecke der Qualitätssicherung und Messtechnik. Mit VGSTUDIO war es ab 1998 erstmals möglich, dass einzelne Schichtbilder nicht mehr isoliert betrachtet werden mussten. Die CT-Scans konnten räumlich zusammenhängend in allen drei Dimensionen auf einem handelsüblichen Personal Computer analysiert und visualisiert werden.

Das Computerprogramm rekonstruiert aus vielen Einzelaufnahmen eines CT-Scanners einen dreidimensionalen Volumendatensatz. Das Ergebnis ist ein exaktes Abbild des realen Objekts, das möglichst alle Merkmale des Originals enthält. Die Voxel werden dabei in Grauwerten dargestellt. Aus den unterschiedlichen Grauwerten der Voxel lassen sich Rückschlüsse auf die Materialbeschaffenheit ziehen (Wikipedia, 2018).

Für die Verwendung von VGSTUDIO MAX ist eine kostspielige Lizenz notwendig. Im Rahmen dieser Arbeit konnte jedoch eine Testlizenz erworben werden, welche für 30 Tage gültig ist. Die Testlizenz ermöglicht es, VGSTUDIO MAX mit allen aktuell verfügbaren Zusatzmodulen auszuprobieren. Um keine Zeit zu verlieren, beschränkte man sich aber auch hier auf die Segmentierung der rechten Niere, welche folgend erläutert wird. Für die vorliegende Arbeit wurde die Version VGSTUDIO MAX 3.2.4 verwendet.

Vorgehen

Auch bei dieser Software wurde der bereits bekannte Bilddatensatz eingelesen. Bereits nach dem Einlesen des Datensatzes erscheinen die Ansichten der unterschiedenen Körperebenen sowie die 3D-Darstellung des Körpers. In Abbildung 14 ist das User Interface von VGSTUDIO MAX mit dem bereits eingelesenen Bilddatensatz zu sehen. Ausserdem wurde die ROI bereits definiert.

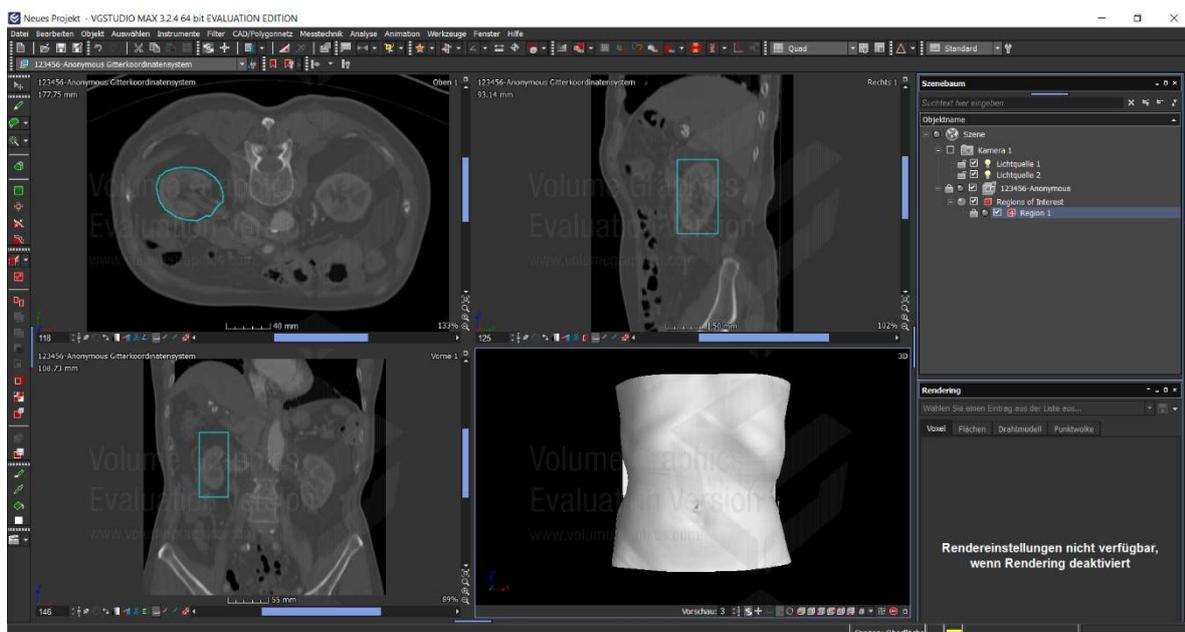


Abbildung 14: User Interface von VGSTUDIO MAX mit bereits eingelesenem Bilddatensatz und definierter ROI

In einem nächsten Schritt wurde sämtliches Gewebe ausserhalb der ROI ausgeblendet (Abbildung 15). Die Fortschritte werden stets unten rechts in der 3D-Visualisierung angezeigt.

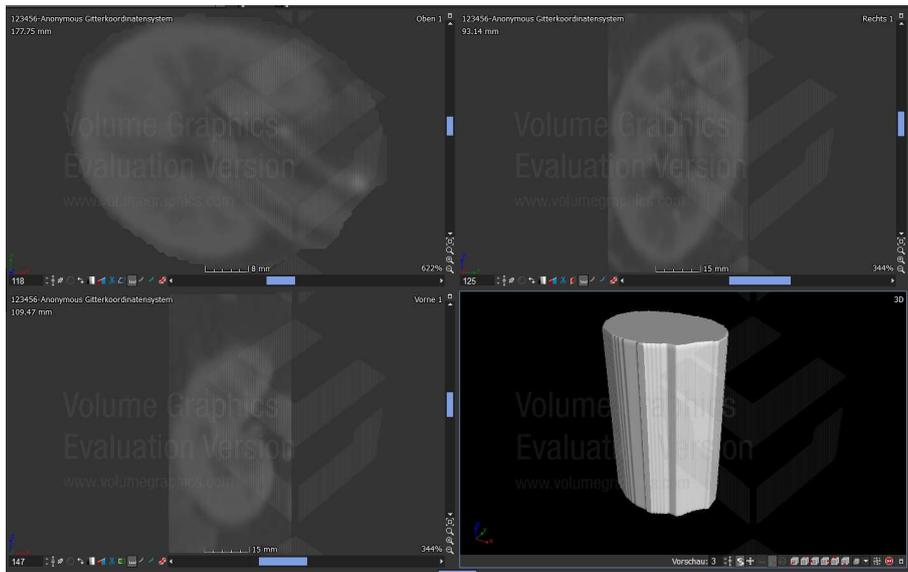


Abbildung 15: Sämtliches Gewebe, welches ausserhalb der ROI liegt, wurde ausgeblendet. Unten rechts erkennt man die ROI als 3D-Visualisierung.

Abschliessend folgt die eigentliche Segmentierung. Dazu wird ein Polygonnetz generiert, welches sämtliches Gewebe oberhalb eines bestimmten Schwellenwert segmentiert. Dieses Verfahren ist also vergleichbar mit dem pixelorientierten Schwellenwertverfahren, welches bereits bei der Software 3D Slicer Anwendung gefunden hat. Der Schwellenwert wurde auch hier auf 50 gesetzt, um das Ergebnis mit den anderen Programmen vergleichbar zu halten (Abbildung 16). Der Vorteil dieses Verfahrens ist, dass sich dieses Polygonnetz anschliessend als STL-Datei exportieren lässt, was die Vergleichbarkeit zu den anderen Programm ebenfalls sicherstellt.

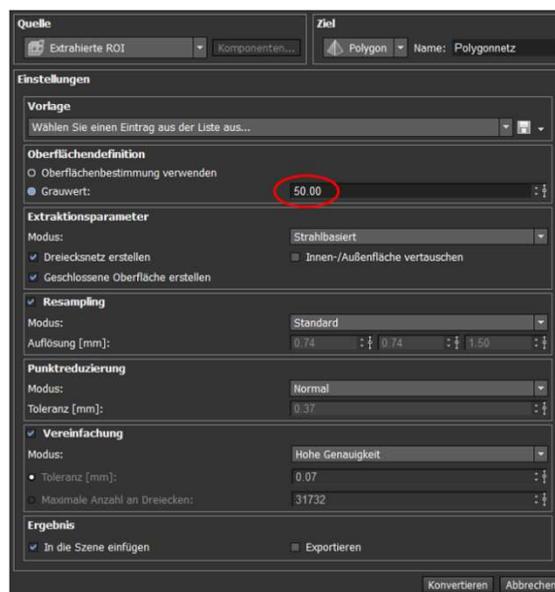


Abbildung 16: Erstellung eine Polygonnetzes mit Schwellenwert 50 (rote Markierung)

2.2 Implementierung eines Segmentierungsprototyps

2.2.1 Prototyp «Alpha»

Bei diesem Prototyp wurde ein Code implementiert, bei welchem man sich unter anderem beim Visualization Toolkit (VTK) bediente. Das VTK ist eine Open-Source-Grafikbibliothek, die sich besonders für die 3D-Computergrafik mit Fokus auf die wissenschaftliche Visualisierung eignet. Sie entstand in einer Forschungsabteilung des Unternehmens General Electric.

Der Code wurde in der Entwicklungsumgebung Spyder 3.2.4 der Freemium-Open-Source-Distribution Anaconda implementiert. Der in Python geschriebene Code liest zunächst eine Reihe von DICOM-Dateien ein, welche mittels CT generiert wurden. Anschliessend wird unter Anwendung des Discrete Marching Cubes Algorithmus die Netzoberfläche eines bestimmten Gewebes aus einem 3D-Volumen extrahiert und visualisiert. Die Extrahierung erfolgt nach dem Prinzip des Schwellenwertverfahrens, wobei der Schwellenwert mittels Hounsfield-Skala bestimmt wird. Schliesslich wird die extrahierte Oberfläche als STL-Datei gespeichert.

In Abbildung 17 ist der Output des Codes abgebildet, links in der Entwicklungsumgebung Spyder und rechts die exportierte STL-Datei, welche mittels 3D-Viewer geöffnet wurde. Der Schwellenwert beträgt 113.



Abbildung 17: Output des ersten Prototyps in der Entwicklungssoftware Spyder (links) und als STL-Export (rechts) mit Schwellenwert 113

Um die Oberfläche zu optimieren, wurde der Code mit dem SmoothPolyDataFilter-Algorithmus ergänzt, dessen Zweck die Glättung (Smoothing) der Oberfläche ist. Der Output des optimierten Codes ist im Kapitel Resultate ersichtlich. Der Nutzen dieses zusätzlichen Algorithmus ist besonders im direkten Vergleich der Oberflächen zu erkennen. Der Glättungsgrad sowie die Anzahl Iterationen lässt sich manuell einstellen, sodass die Oberfläche individuell an die Anforderungen angepasst werden kann.

2.2.2 Prototyp «Beta»

Da durch den Glättungs-Algorithmus aus dem vorher beschriebenen Prototyp die Gefahr besteht, dass Konturen oder Formen im 3D-Modell verschwinden könnten - da sie quasi «weggeglättet» werden - wurde ein zweiter Prototyp implementiert, der bereits bei der Segmentierung genauer vorgeht und so eine detailliertere Oberfläche generiert. Damit soll verhindert werden, dass im segmentierten Modell wichtige Strukturen ausgeblendet oder entfernt werden. Somit wird eine möglichst der Realität entsprechende Abbildung sichergestellt.

Bei diesem Prototyp bediente man sich unter anderem in der Programmbibliothek Matplotlib. Auch dieser Code liest zunächst eine Reihe von DICOM-Dateien aus dem eingegebenen Verzeichnis ein. Dabei wird die Anzahl aller sowie der Pfadname der ersten fünf Dateien ausgegeben (Abbildung 18). Damit stellt man sicher, dass die Dateien aus dem richtigen Verzeichnis eingelesen wurden.

```
Total of 215 DICOM images.  
First 5 filenames:  
C:\Users\tbvonwyl\Documents\BAT\Datenstets\DICOM-Body\ScalarVolume_16\IMG0001.dcm  
C:\Users\tbvonwyl\Documents\BAT\Datenstets\DICOM-Body\ScalarVolume_16\IMG0002.dcm  
C:\Users\tbvonwyl\Documents\BAT\Datenstets\DICOM-Body\ScalarVolume_16\IMG0003.dcm  
C:\Users\tbvonwyl\Documents\BAT\Datenstets\DICOM-Body\ScalarVolume_16\IMG0004.dcm  
C:\Users\tbvonwyl\Documents\BAT\Datenstets\DICOM-Body\ScalarVolume_16\IMG0005.dcm
```

Abbildung 18: Die Anzahl der eingelesenen sowie der Pfadname der ersten fünf Dateien wird direkt im Programm ausgegeben.

Anschliessend werden die Rohdaten der Voxelwerte des gesamten Datensatzes in die Hounsfield-Einheiten konvertiert, welche schliesslich in Form eines Histogramms ausgegeben werden. Dieses ist in Abbildung 19 dargestellt. Aus dem Histogramm lässt sich ableiten, welches Gewebe zu welchen Anteilen im eingelesenen Datensatz vorkommt.

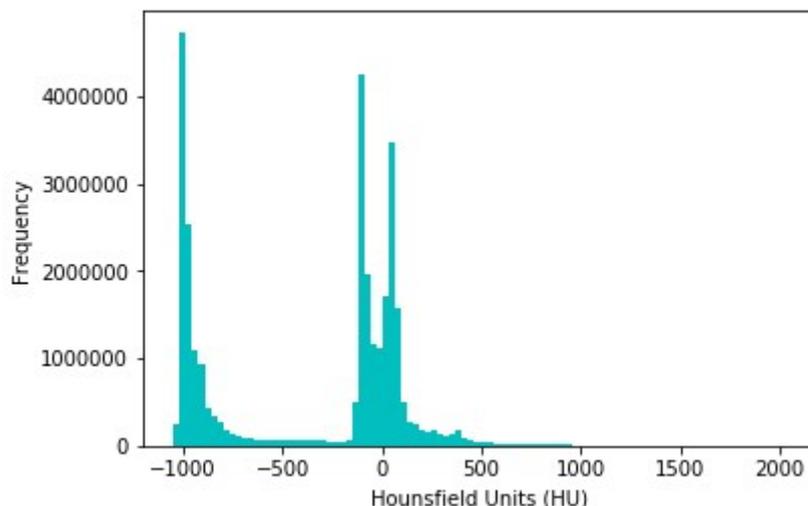


Abbildung 19: ausgegebenes Histogramm auf Basis des eingelesenen Bilddatensatzes

Nachfolgend erscheint der eingelesene Bildstapel, wobei auf Grund der limitierten Bildschirmgröße in diesem Beispiel nur jedes achte Bild in einem 5x5 Raster angezeigt wird, beginnend mit dem ersten Bild (slice 0) (Abbildung 20). Auch dieser Schritt dient lediglich zur Kontrolle, dass der richtige Datensatz eingelesen wurde und hat somit keinen Einfluss auf die anschließende Segmentierung, bei welcher sämtliche Bilddaten des Bilddatensatzes berücksichtigt werden.

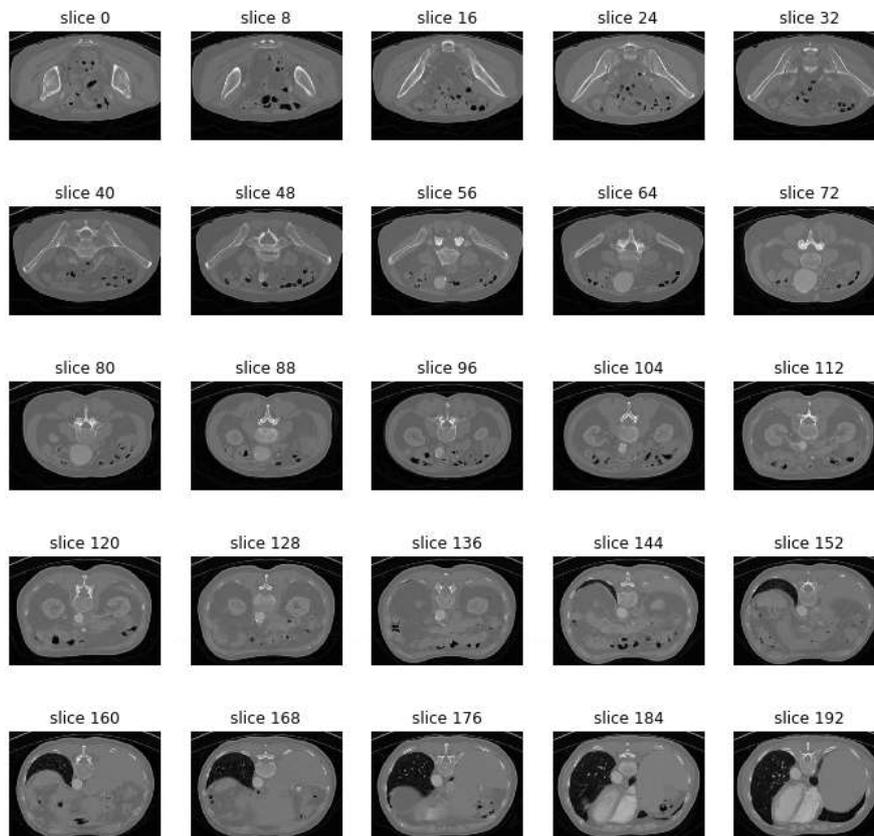


Abbildung 20: Darstellungen der eingelesenen Bilddaten, wobei nur jede achte Abbildung beginnend mit der ersten Abbildung (slice 0) angezeigt wird

In einem nächsten Schritt wird die Schichtdicke sowie die Voxelgröße analysiert. Diese Informationen entnimmt der Code aus den DICOM Headinformationen des Datensatzes und gibt diese in der Entwicklungsumgebung aus (Abbildung 21). Beim eingelesenen Datensatz beträgt die Schichtdicke 1.50 mm und ein Voxel entspricht 0.742x0.742 mm. Das nachfolgende Resampling stellt die richtige Skalierung sicher, damit das segmentierte Modell die richtigen Größenverhältnisse aufweist. In diesem Schritt wird schlussendlich auch die Auflösung des Modells festgelegt. In diesem Beispiel beträgt die Auflösung 1x1x1 mm. Die Fortschritte des Resamplings werden ebenfalls ausgegeben (Abbildung 21).

```
Slice Thickness: 1.500000
Pixel Spacing (row, col): (0.742188, 0.742188)
Shape before resampling (215, 321, 441)
Shape after resampling (322, 238, 327)
```

Abbildung 21: Angabe der Schichtdicke, der Voxelgröße sowie die Angaben zum Bilddatensatz vor und nach dem Resampling

Anschließend wird bei diesem Prototyp der Classic Marching Cubes Algorithmus angewendet, um die Netzoberfläche eines bestimmten Gewebes aus einem 3D-Volumen zu extrahieren. Auch dieser Algorithmus basiert auf dem Schwellenwertverfahren. Die extrahierte Oberfläche wird sogleich in der Entwicklerumgebung angezeigt. Zudem wird im selben Schritt eine STL-Datei der Oberfläche generiert und diese in ein gewünschtes Verzeichnis exportiert.

Die Darstellung der extrahierten Oberfläche in der Entwicklungsumgebung sowie als STL-Export ist in Abbildung 22 zu sehen.

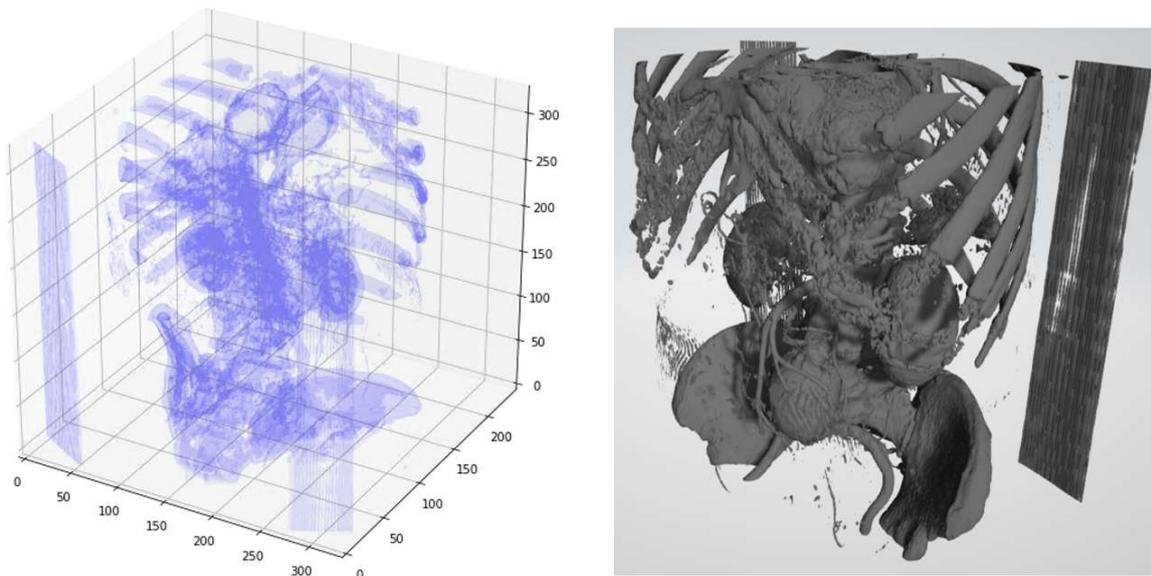


Abbildung 22: extrahierte Oberfläche des zweiten Prototyps, dargestellt in der Entwicklungsumgebung (links) und als STL-Export (rechts)

Wie bereits angesprochen, wurde beim Resampling die Auflösung auf 1x1x1 mm eingestellt. Diese Auflösung kann manuell geändert werden, was besonders Einfluss auf die Beschaffenheit der extrahierten Oberfläche hat. Die Auflösung wird nun von 1x1x1 mm auf 0.4x0.4x0.4 mm geändert. Abbildung 23 zeigt das daraus resultierende Modell als STL-Export.

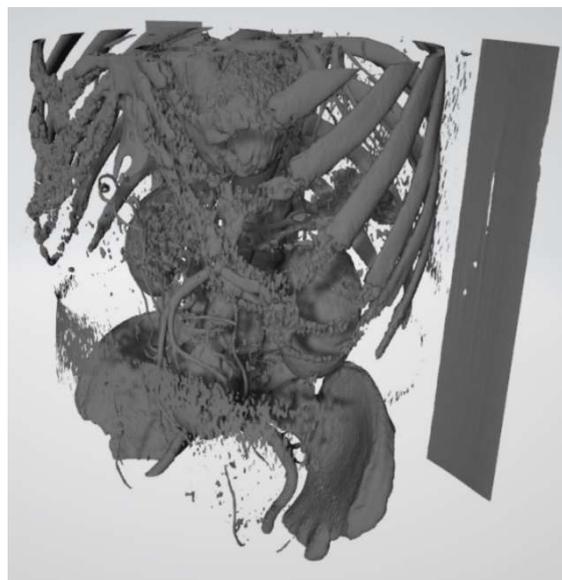


Abbildung 23: STL-Export mit Auflösung 0.4x0.4x0.4 mm

Wie so oft sind die Unterschiede am besten im direkten Vergleich zu erkennen. In Abbildung 24 wird anhand eines Ausschnitts der linken Niere die unterschiedliche Oberflächenbeschaffenheit deutlich gemacht.

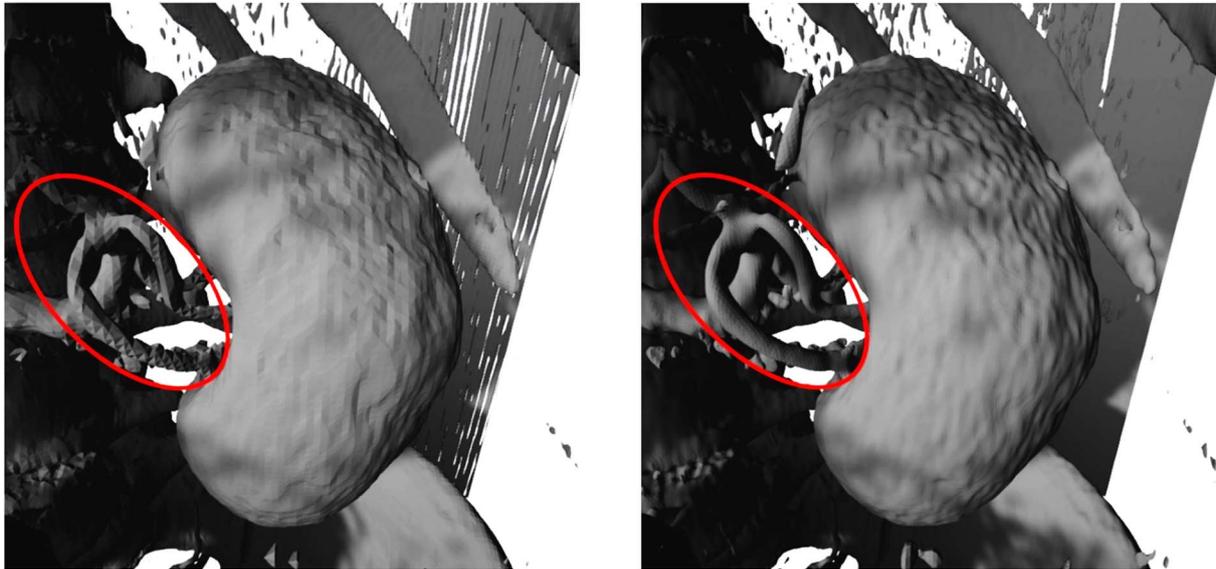


Abbildung 24: Vergleich der Oberflächenbeschaffenheit: Auflösung 1x1x1 mm(links) und Auflösung 0.4x0.4x0.4 mm(rechts). Besonders deutlich werden die Unterschiede bei der Betrachtung der Nierenarterie und -vene (rote Markierung).

2.3 Überprüfung/Validierung des Segmentierungsprototyps

2.3.1 Voraussetzungen

Um die Segmentierungsprototypen nachvollziehbar zu testen und diese Validierung auch vergleichbar zu gestalten, müssen zunächst die Prüfkriterien definiert werden. Ausserdem gilt es, die Voraussetzungen für die Validierung zu beschreiben. Geprüft wird ausschliesslich der STL-Export der Prototypen, da somit dieselben Ausgangslagen geschaffen werden können. Zum Betrachten des Modells wird der 3D-Viewer von Microsoft verwendet. Als Kriterien für die Validierung wurden folgende Prüfkriterien festgelegt:

1. Geschwindigkeit

Die Zeit, die verstreicht, vom Zeitpunkt des Segmentierungsstarts bis zum lokal abgespeicherten STL-Export.

2. Grösse des STL-Exports

Dateigrösse der STL-Datei des Modells, welche am Speicherort angezeigt wird.

3. Detailtreue des Modells

Prüft, ob feine Strukturen auf dem Modell ersichtlich sind.

4. Oberflächenglätte des Modells

Untersucht die Oberflächenbeschaffenheit des Modells.

Die Prüfkriterien «Geschwindigkeit» und «Grösse des STL-Exports» sind objektiv zu beurteilen und können direkt als konkreten Wert angegeben werden. Die Kriterien «Detailtreue des Modells» und «Oberflächenglätte des Modells» sind subjektiv zu beurteilen. Dazu wurde für die entsprechenden Kriterien folgende Bewertungstabelle erstellt:

Detailtreue des Modells		
1	schlecht	Mittelgrosse Strukturen sind im Modell nicht zu erkennen.
2	eher schlecht	Feine Strukturen sind im Modell nicht zu erkennen.
3	gering	Feine Strukturen sind im Modell ansatzweise zu erkennen.
4	eher gut	Feine Strukturen sind im Modell abgebildet, jedoch nur grosszügig.
5	gut	Feine Strukturen sind im Modell abgebildet, jedoch nicht detailliert.
6	hervorragend	Feine Strukturen sind im Modell detailliert und realistisch abgebildet.

Oberflächenglätte des Modells		
1	schlecht	Durch die kantige Oberfläche werden Strukturen nicht dargestellt.
2	eher schlecht	Die Oberfläche wird kantig und/oder rau dargestellt und entspricht nicht der Realität.
3	gering	Die Oberfläche wirkt kantig und/oder rau.
4	eher gut	Die Oberfläche wirkt unnatürlich.
5	gut	Die Oberfläche wird realistisch dargestellt und wirkt nur bei genauer Betrachtung unnatürlich.
6	hervorragend	Die Oberfläche wird realistisch und natürlich dargestellt.

Als Referenz für die Bewertung wird die Segmentierung zusätzlich mit der Industrie-Software VGSTUDIO MAX und der freien Software 3D Slicer durchgeführt. Diese Massnahme soll dazu beitragen, das qualitative Niveau der Prototypen einschätzen zu können.

Für die Testfälle wurde ein Rechner mit folgenden Spezifikationen verwendet:

Prozessor: Intel® Core™ i5-6200U CPU @ 2.30GHz 2.40 GHz

RAM: 8.00 GB

2.3.2 Durchführung

Während des Tests werden drei Testfälle betrachtet. Für jeden Testfall wurde ein eigener Bilddatensatz verwendet, der jeweils eine andere Segmentierung zum Ziel hat. Bei jedem Testfall werden zunächst Informationen zum Bilddatensatz aufgelistet. Weiter werden die Einstellungen bei der Segmentierung des jeweiligen Programms beschrieben. Anschliessend folgt die Beschreibung der Ergebnisse aus der Segmentierung, welche schliesslich einer Bewertung nach den oben genannten Kriterien unterzogen werden.

3 Resultate

In diesem Kapitel werden zunächst die Ergebnisse und die Auswertung der im vorherigen Kapitel beschriebenen Analyse der Segmentierungsapplikationen vorgestellt. Anschliessend folgen die Resultate des SmoothPolyDataFilter-Algorithmus im Prototyp «Alpha» sowie die Ergebnisse der Überprüfung und die Validierung der beiden Prototypen «Alpha» und «Beta».

3.1 Analyse der Segmentierung-Programme

3.1.1 3D Slicer

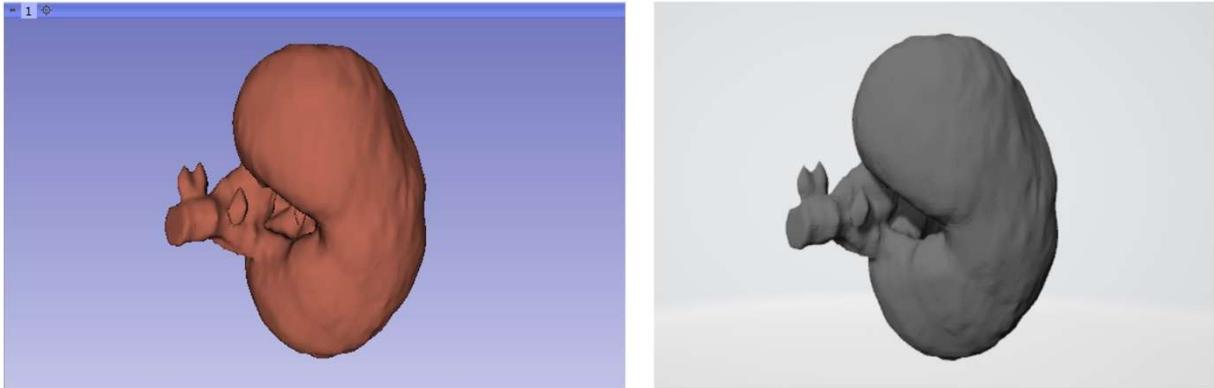


Abbildung 25: 3D-Visualisierung (links) sowie der STL-Export (rechts) der segmentierten Niere aus 3D Slicer

Die Analyse von 3D Slicer hat gezeigt, dass unterschiedliche Algorithmen zur Verfügung stehen, um eine Segmentierung durchzuführen. Für das durchgespielte Beispiel reichte ein einfaches Schwellenwertverfahren, in Kombination mit Erosions- und Dilatations-Algorithmen, um ein Ergebnis zu erhalten. Kenntnisse über solche Algorithmen müssen dem Benutzer aber vorher bekannt sein.

Was die Segmentierung etwas erschwerte, war die Tatsache, dass die 3D-Ansicht während der Schwellenwertanpassung nicht in Echtzeit realisiert werden konnte. So mussten mehrere Versuche unternommen werden, um den passenden Schwellenwert zu finden.

3.1.2 ITK-SNAP

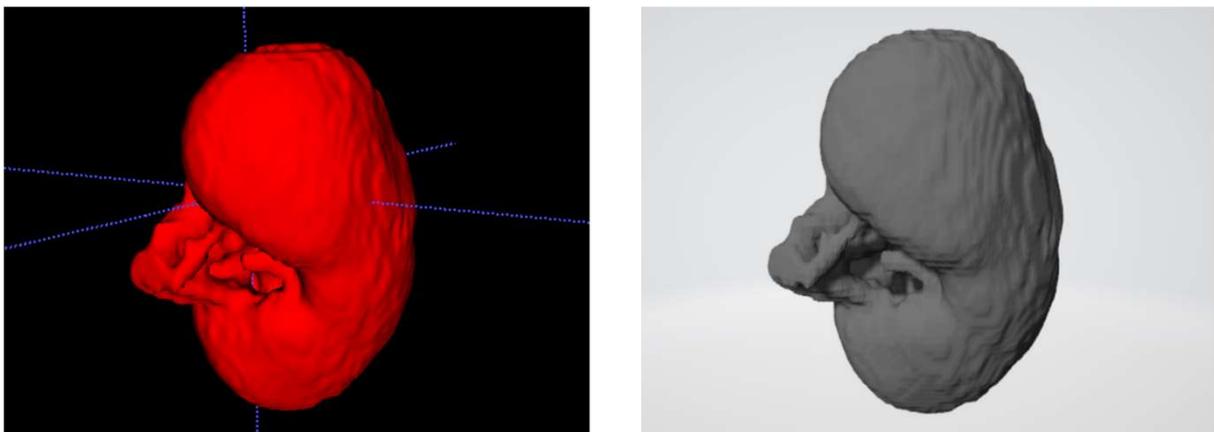


Abbildung 26: 3D-Visualisierung (links) sowie der STL-Export (rechts) der segmentierten Niere aus ITK-SNAP

Auch durch diese Software ist es gelungen, die Niere zu segmentieren. Dazu wurde jedoch ein anderes Verfahren als bei 3D Slicer verwendet. Der Segmentierungsprozess wird ausserdem in mehrere Schritte unterteilt, was eine intuitive Bedienung gewährleistet. Eine aufwändige Einarbeitung in das Programm ist dadurch nicht notwendig.

Was die Segmentierung auch hier erschwerte war die nicht vorhandene Echtzeitvisualisierung in 3D der Schwellenwertsänderung. Zudem lassen sich die Schwellenwerte nicht einfach ändern, wenn man bereits beim Region Growing angelangt ist. Möchte man die Schwellenwerte zu diesem Zeitpunkt noch ändern, muss man erneut mit dem ersten Schritt beginnen, während sämtliche Fortschritte gelöscht werden.

3.1.3 VGSTUDIO MAX

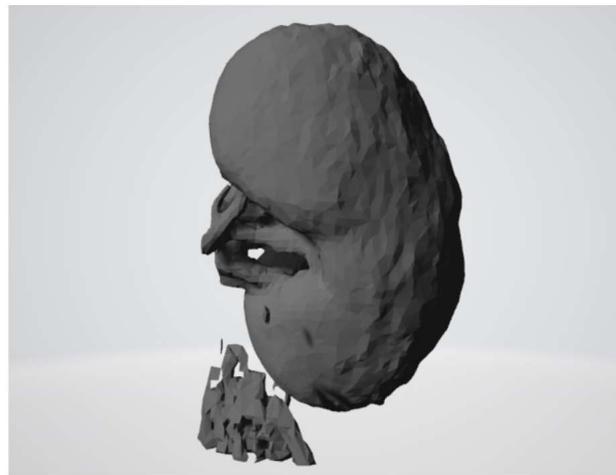


Abbildung 27: 3D-Visualisierung (links) sowie der STL-Export (rechts) der segmentierten Niere aus VGSTUDIO MAX

Die Segmentierung der Niere ist auch hier gelungen. Das verwendete Verfahren entspricht dem pixelorientierten Schwellenwertverfahren, da auch hier ein Schwellenwert gesetzt wurde, auf Grund dessen anschliessend ein Polygonnetz erstellt wurde. Wie man anhand des Ergebnisses erkennen kann, sind nicht zusammenhängende Strukturen entstanden, was ein deutlicher Nachteil dieses Verfahrens darstellt. Dies hätte durch eine Kombination von Erosions- und Dilatations- Algorithmen vermieden werden können. Die Funktion dieser Algorithmen in dieser Software konnten aber während der Segmentierung nicht richtig nachvollzogen werden. Aus diesem Grund hat man darauf verzichtet. Allgemein kann man sagen, dass die Bedienung von VGSTUDIO MAX eher umständlich und nicht intuitiv ist. Dies ist jedoch als subjektive Wahrnehmung zu werten und hängt bestimmt auch damit zusammen, dass die Erfahrung in der Verwendung dieses Programms fehlt.

Auch beim Ergebnis der Segmentierung lassen sich Auffälligkeiten feststellen. So sind zwar im Modell im Vergleich zu den anderen Programmen mehr Details zu erkennen, die Oberfläche wirkt aber rau und kantig.

3.1.4 Fazit

Die Segmentierung der Niere mit unterschiedlichen Programmen hat zu wichtigen Erkenntnissen geführt. So kann man sich mit der Wahl des richtigen Segmentierungsverfahrens viel Arbeit sparen. Wählt man beispielsweise ein regionenorientiertes Segmentierungsverfahren, erhält man direkt eine zusammenhängende Struktur, ohne anschliessend Erosions- und Dilatations-Algorithmen anzuwenden, wie das bei pixelorientierten Verfahren häufig der Fall ist.

Wichtige Unterschiede lassen sich auch in der Qualität des Modells erkennen. Beim Modell von VGSTUDIO MAX ist die Detailtreue deutlich höher als bei den anderen beiden Modellen. Dafür überzeugt die Oberflächenglätte beim Modell des 3D Slicers am meisten. Diesen beiden Faktoren wurden auch bei der Implementierung der eigenen Prototypen besondere Beachtung geschenkt. Auch die Rechenleistung und somit die Geschwindigkeit des Programms ist ein Faktor, an dem es stets etwas zu optimieren gibt.

3.2 Resultate des SmoothPolyDataFilter-Algorithmus

Wie bereits erwähnt, wurde der Prototyp «Alpha» mit dem SmoothPolyDateFilter-Algorithmus ergänzt, welcher den Zweck hat, die Oberfläche zu glätten (Smoothing). In den Abbildungen 28, 29 und 30 sind die Auswirkungen dieses Glättung-Algorithmus im direkten Vergleich zu erkennen. Der Glättungsgrad sowie die Anzahl Iterationen sind manuell einzustellen und wurden in diesem Beispiel auf Glättungsfaktor = 0.3 und Anzahl Iterationen = 15 festgelegt.

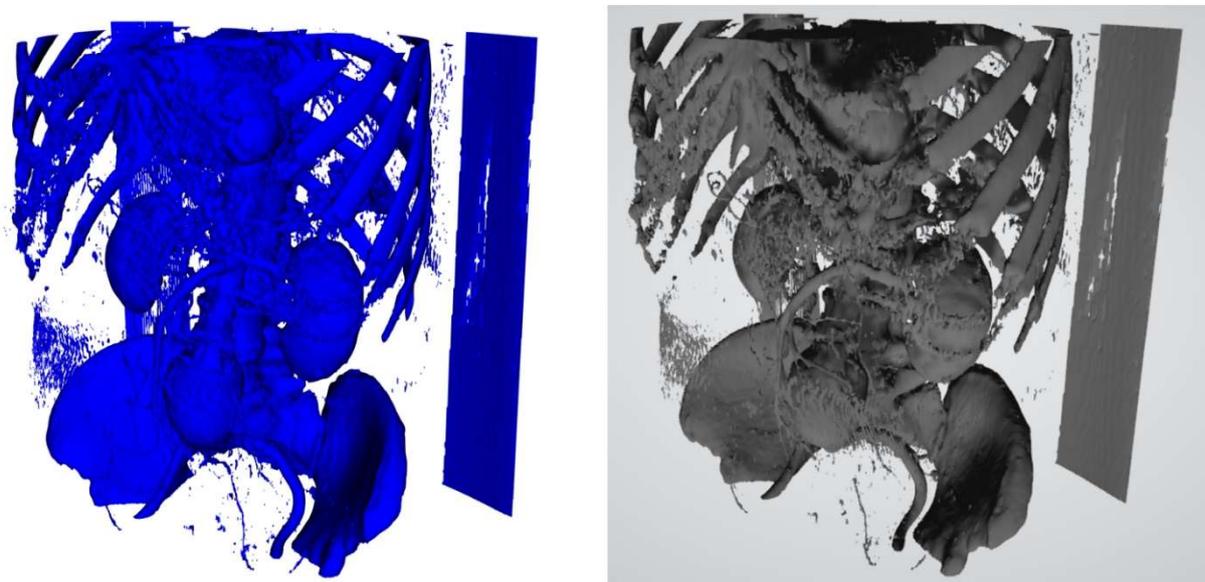


Abbildung 28: Output des Prototyps «Alpha» in der Entwicklungssoftware Spyder (links) und als STL-Export (rechts) mit Glättungs-Algorithmus mit Anzahl Iterationen = 15 und Glättungsfaktor 0.3, Schwellenwert = 113

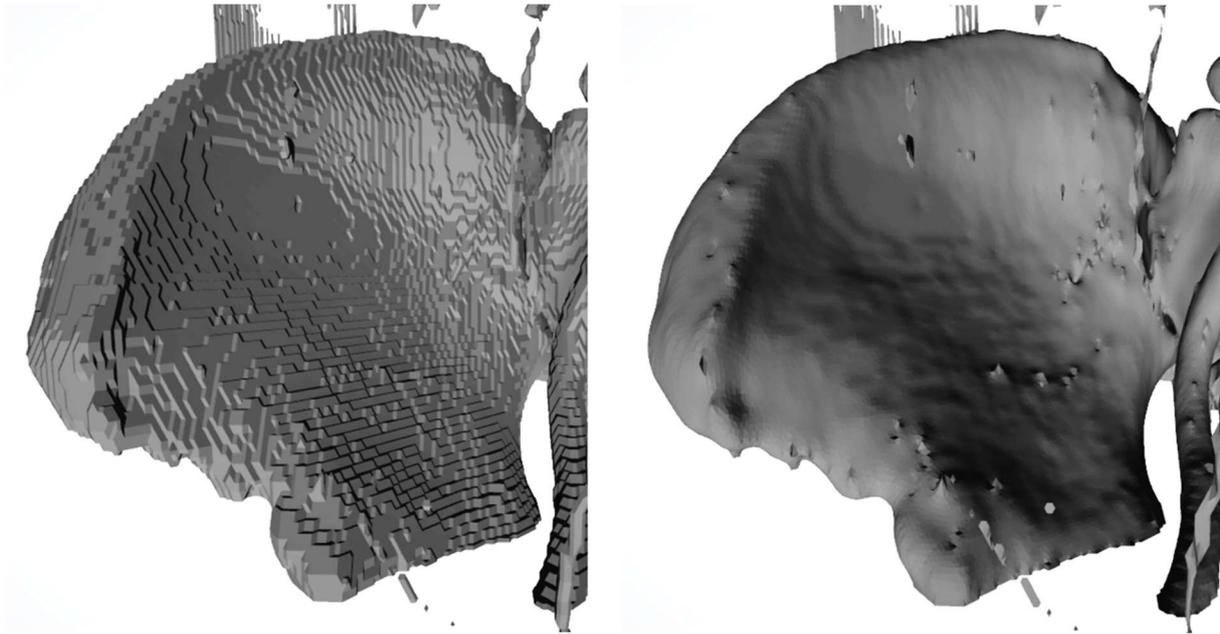


Abbildung 29: Ausschnitt des rechten Beckenknochens im STL-Export ohne Glättungs-Algorithmus (links) und mit Glättungs-Algorithmus mit Anzahl Iterationen = 15 und Glättungsfaktor 0.3 (rechts), Schwellenwert = 113

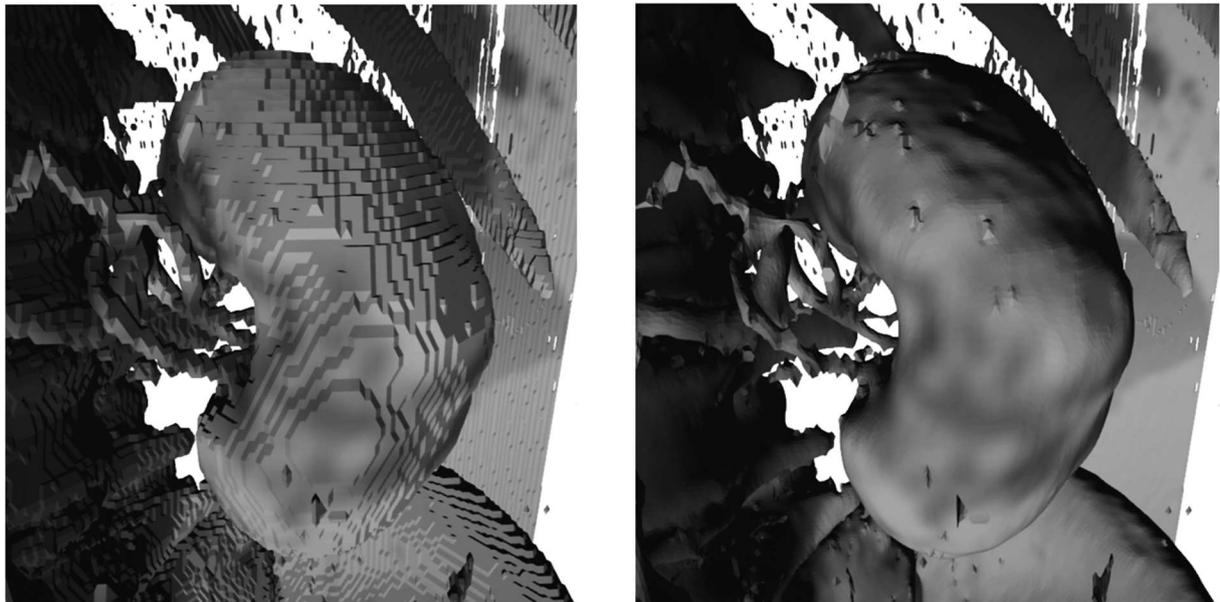


Abbildung 30: Ausschnitt der linken Niere im STL-Export ohne Glättungs-Algorithmus (links) und mit Glättungs-Algorithmus mit Anzahl Iterationen = 15 und Glättungsfaktor 0.3 (rechts), Schwellenwert = 113

3.3 Validierung

3.3.1 Testfall 1 – «Thorax»

Informationen zum Bilddatensatz	
Abgebildeter Körper	Rumpf
Anzahl Bilder	215
Datei-Format	DICOM

Informationen zur Segmentierung				
Ziel	Segmentierung der Beckenknochen und der Nieren			
Verfahren	pixelorientiertes Schwellenwertverfahren			
Schwellenwert	113			
Programm	Prototyp «Alpha»	Prototyp «Beta»	3D Slicer	VGSTUIDO MAX
eingestelltes Smoothing	Iterationen: 15 Glättungsfaktor: 0.3	Kein Smoothing	Kein Smoothing	Kein Smoothing
eingestellte Auflösung in mm	Auflösung nicht einstellbar	0.4x0.4x0.4	Auflösung nicht einstellbar	0.4x0.4x0.4

Resultate

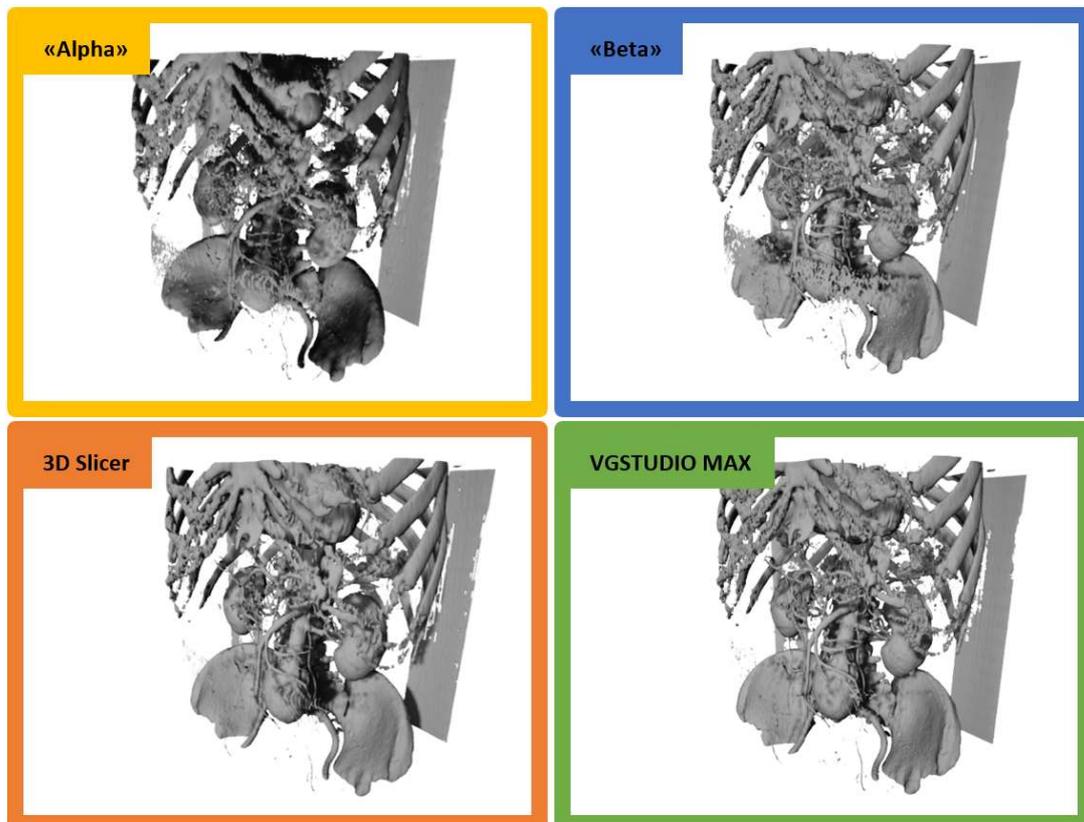


Abbildung 31: Übersicht des segmentierten Datensatzes "Thorax"

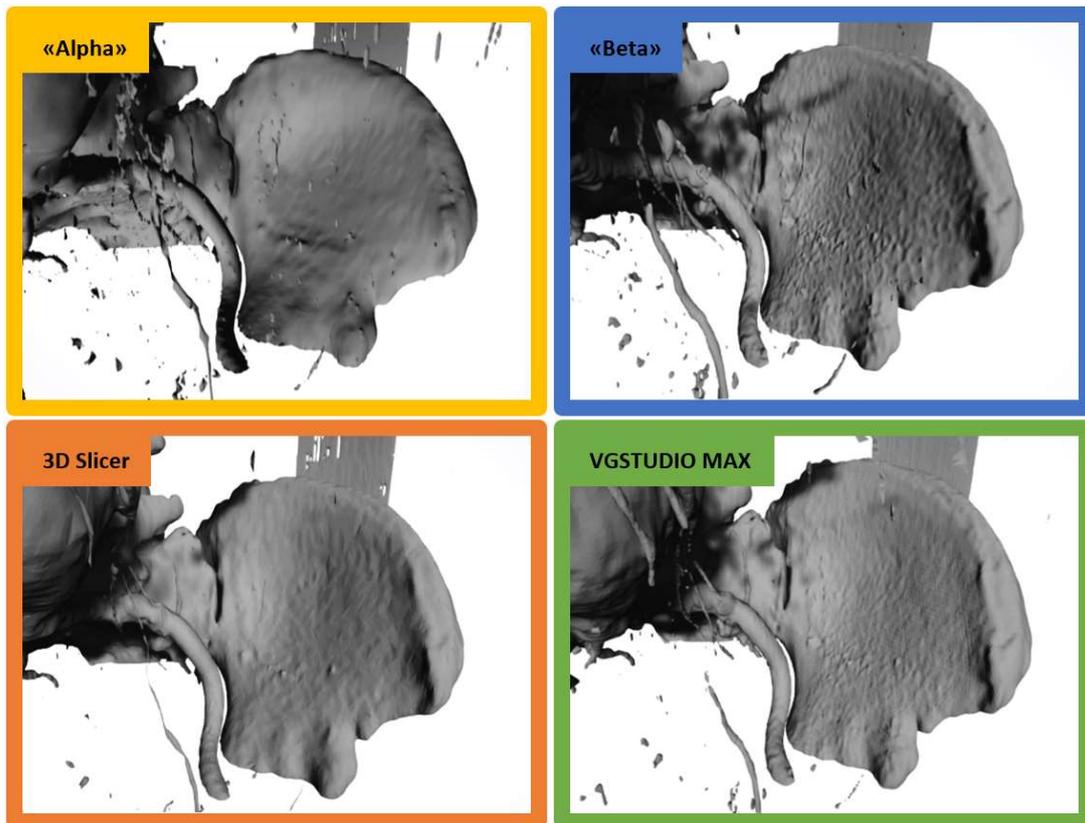


Abbildung 32: Detailansicht des linken Beckenknochens

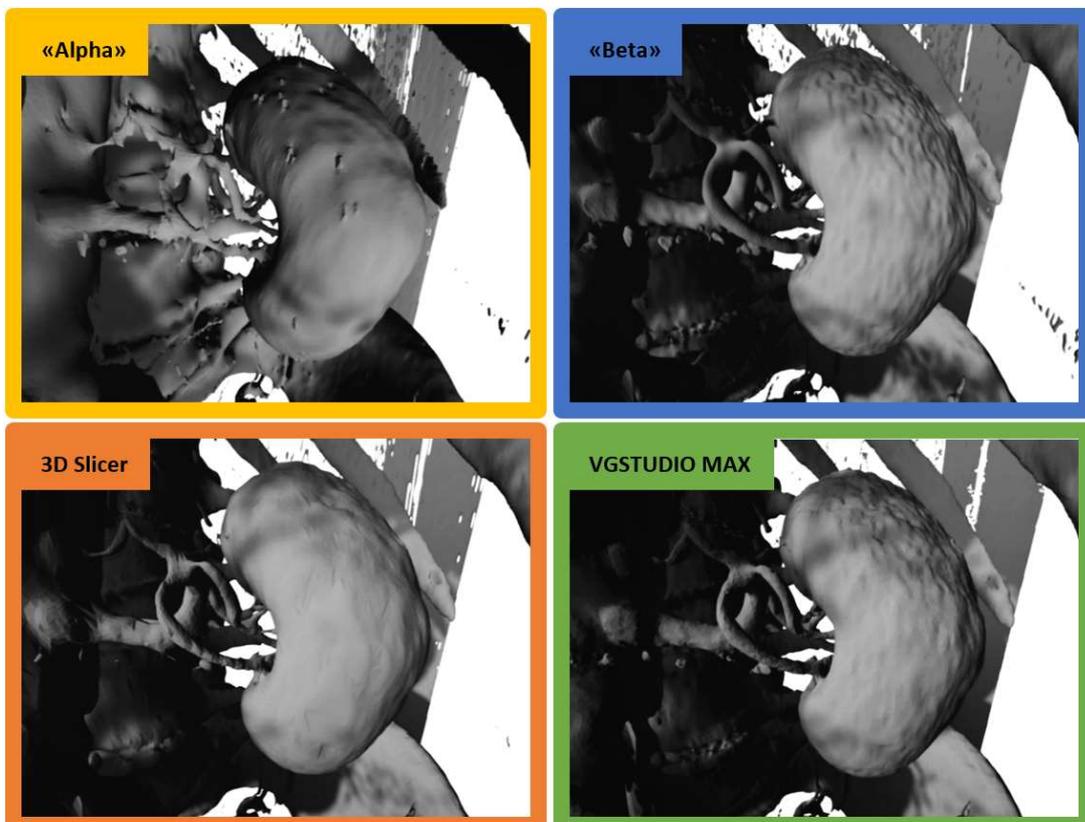


Abbildung 33: Detailansicht der linken Niere

Bewertung

Prototyp «Alpha»	
Geschwindigkeit	00:18 min
Grösse des STL-Exports	120'212 KB
Detailtreue des Modells	2
Oberflächenglätte des Modells	4

Prototyp «Beta»	
Geschwindigkeit	5:08 min*
Grösse des STL-Exports	678'318 KB
Detailtreue des Modells	6
Oberflächenglätte des Modells	6

3D Slicer	
Geschwindigkeit	00:27 min
Grösse des STL-Exports	93'222 KB
Detailtreue des Modells	5
Oberflächenglätte des Modells	4

VGSTUDIO MAX	
Geschwindigkeit	08:07 min
Grösse des STL-Exports	189'490 KB
Detailtreue des Modells	6
Oberflächenglätte des Modells	5

*Da der verwendete Rechner einen relativ kleinen Arbeitsspeicher aufweist, wurde der Prototyp «Beta» an einem leistungsstärkeren Rechner getestet. Dieser Rechner weist folgende Spezifikationen auf:

Prozessor: Intel® Xeon® CPU E5-1650 v4 @ 3.60GHz 3.60GHz

RAM: 128.00 GB

Daher ist die angegebene Zeit nicht mit den anderen Zeiten zu vergleichen.

3.3.2 Testfall 2 – «Truncus»

Informationen zum Bilddatensatz	
Abgebildeter Körper	Rumpf
Anzahl Bilder	215
Datei-Format	DICOM

Informationen zur Segmentierung				
Ziel	Segmentierung des Rumpfs			
Verfahren	pixelorientiertes Schwellenwertverfahren			
Schwellenwert	-300			
Programm	Prototyp «Alpha»	Prototyp «Beta»	3D Slicer	VGSTUIDO MAX
eingestelltes Smoothing	Iterationen: 15 Glättungsfaktor: 0.3	Kein Smoothing	Kein Smoothing	Kein Smoothing
eingestellte Auflösung in mm	Auflösung nicht einstellbar	0.3x0.3x0.3	Auflösung nicht einstellbar	0.3x0.3x0.3

Resultate

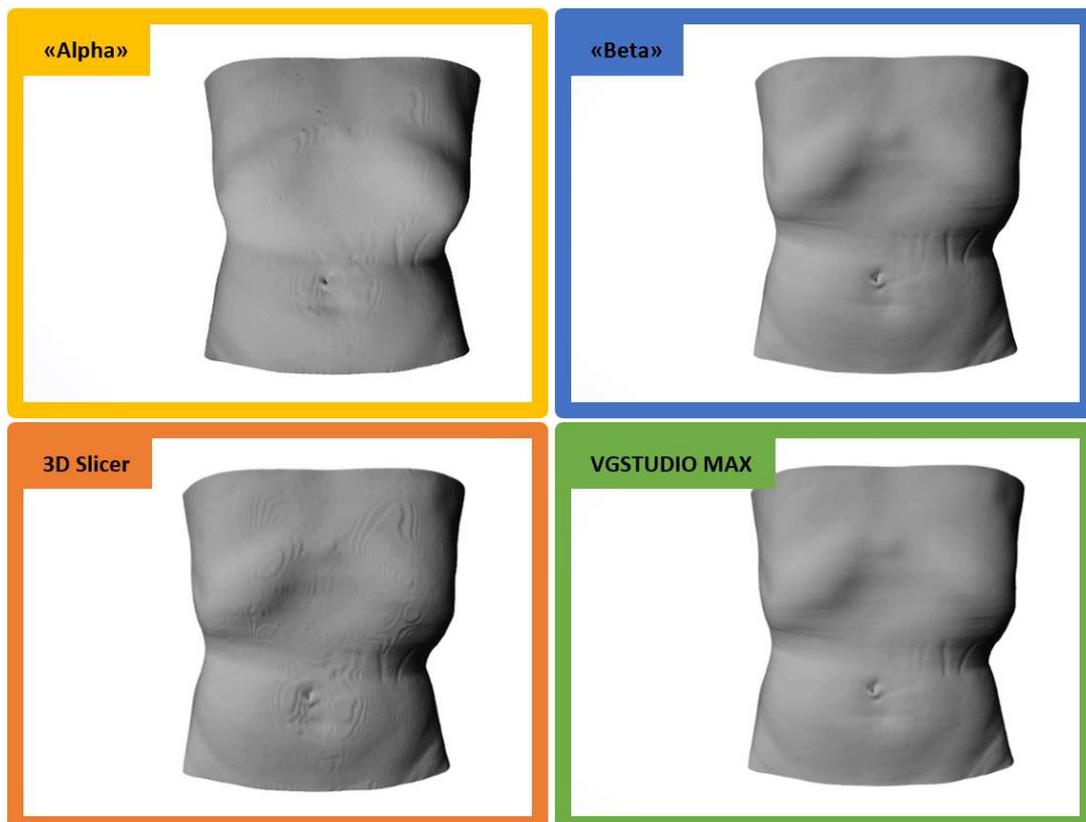


Abbildung 34: Übersicht des segmentierten Datensatzes "Truncus"

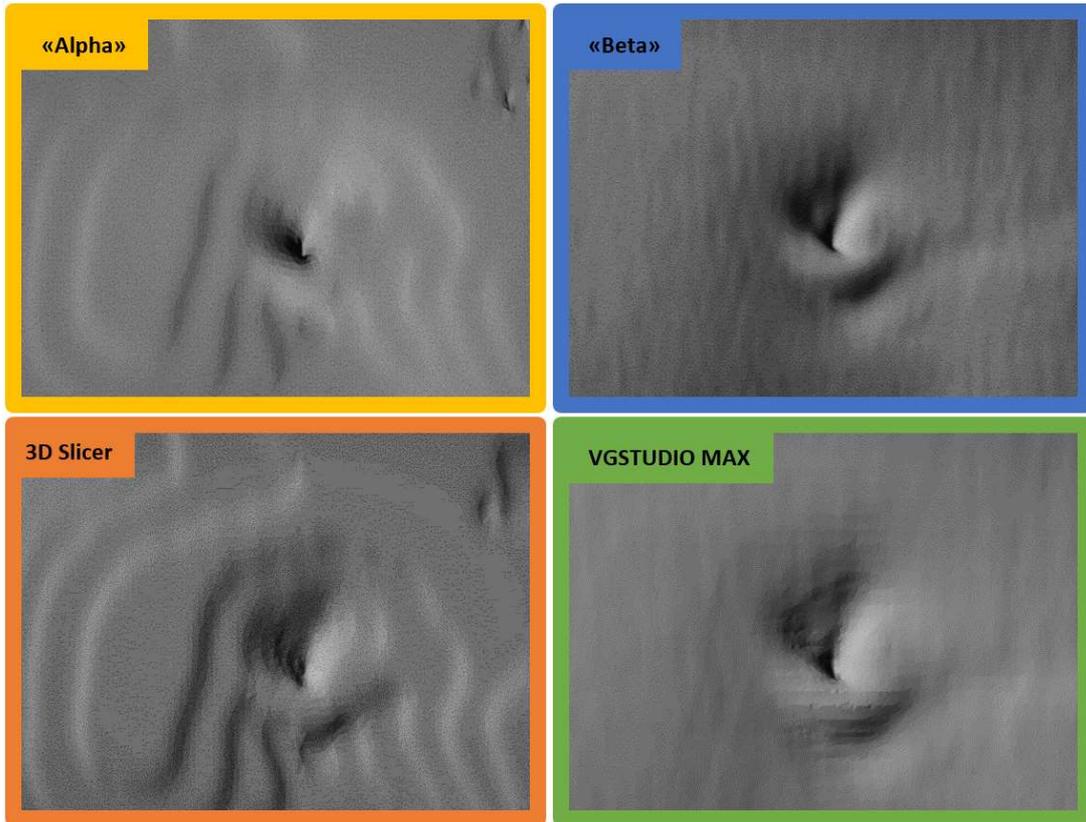


Abbildung 35: Detailansicht des Bauchnabels

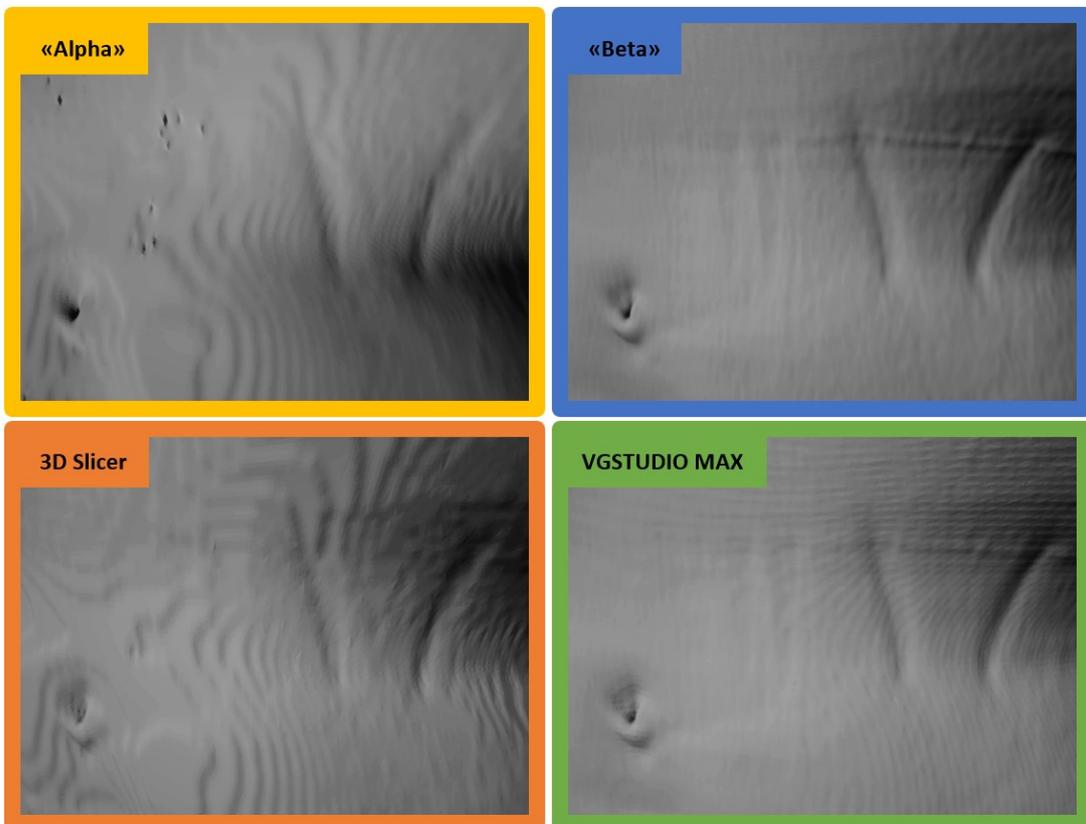


Abbildung 36: Detailansicht der Taille

Bewertung

Prototyp «Alpha»	
Geschwindigkeit	00:15 min
Grösse des STL-Exports	81'318 KB
Detailtreue des Modells	4
Oberflächenglätte des Modells	3

Prototyp «Alpha»	
Geschwindigkeit	9:38 min*
Grösse des STL-Exports	869'678 KB
Detailtreue des Modells	6
Oberflächenglätte des Modells	6

3D Slicer	
Geschwindigkeit	00:19 min
Grösse des STL-Exports	73'427 KB
Detailtreue des Modells	4
Oberflächenglätte des Modells	2

VGSTUDIO MAX	
Geschwindigkeit	10:56 min
Grösse des STL-Exports	257'745 KB
Detailtreue des Modells	6
Oberflächenglätte des Modells	5

*Da der verwendete Rechner einen relativ kleinen Arbeitsspeicher aufweist, wurde der Prototyp «Beta» an einem leistungsstärkeren Rechner getestet. Dieser Rechner weist folgende Spezifikationen auf:

Prozessor: Intel® Xeon® CPU E5-1650 v4 @ 3.60GHz 3.60GHz
RAM: 128.00 GB

Daher ist die angegebene Zeit nicht mit den anderen Zeiten zu vergleichen.

3.3.3 Testfall 3 – «Cranium»

Informationen zum Bilddatensatz	
Abgebildeter Körper	Kopf
Anzahl Bilder	139
Datei-Format	DICOM

Informationen zur Segmentierung				
Ziel	Segmentierung der Schädel- und Kieferknochen			
Verfahren	pixelorientiertes Schwellenwertverfahren			
Schwellenwert	290			
Programm	Prototyp «Alpha»	Prototyp «Beta»	3D Slicer	VGSTUIDO MAX
eingestelltes Smoothing	Iterationen: 15 Glättungsfaktor: 0.3	Kein Smoothing	Kein Smoothing	Kein Smoothing
eingestellte Auflösung in mm	Auflösung nicht einstellbar	0.3x0.3x0.3	Auflösung nicht einstellbar	0.3x0.3x0.3

Resultate

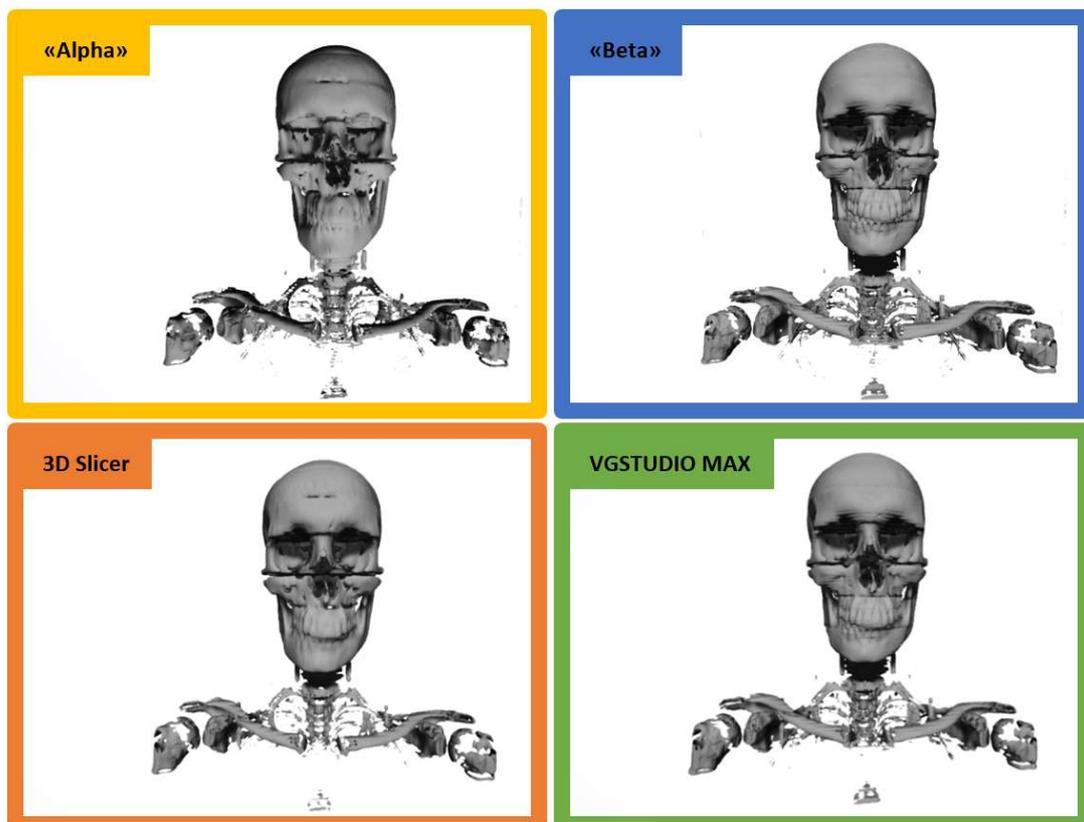


Abbildung 37: Übersicht des segmentierten Datensatzes "Cranium". Die Schnitte im Bereich der Stirn, Augenhöhle, des Jochbeins und des Kiefers sind bereits im Datensatz vorhanden und nicht durch die Segmentierung entstanden.

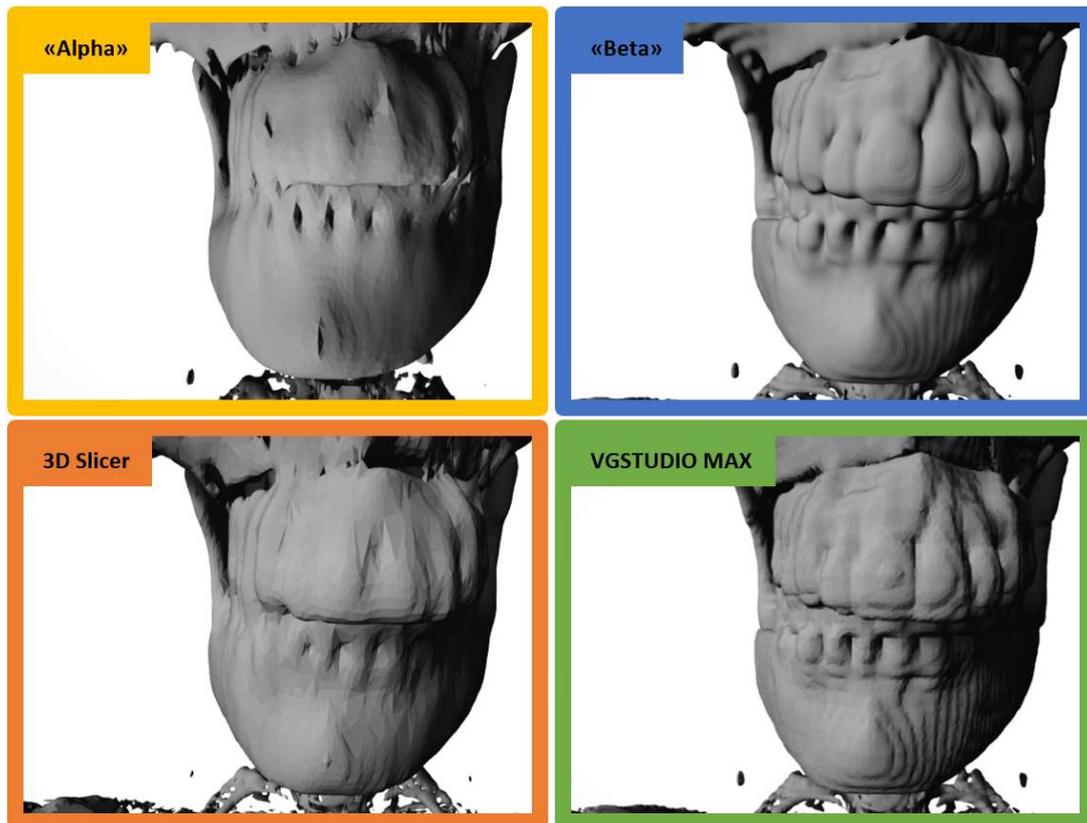


Abbildung 38: Detailansicht des Kiefers und der Zähne

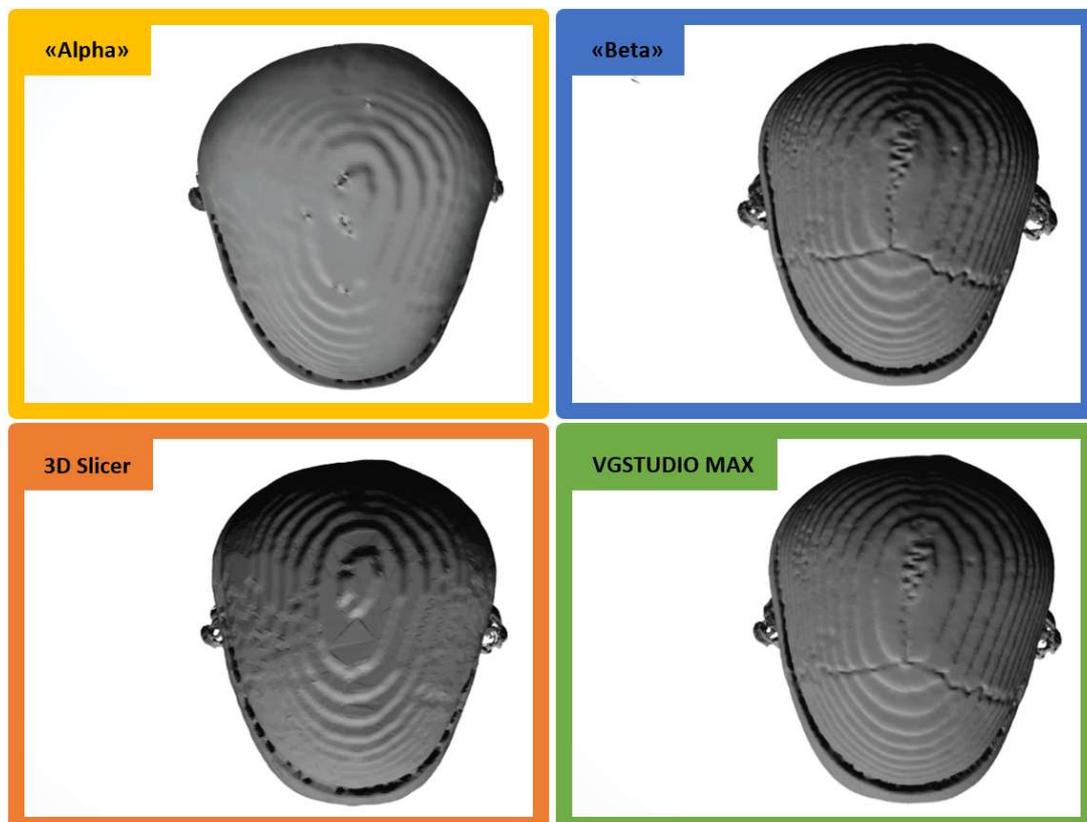


Abbildung 39: Detailansicht der Schädeldecke: Der treppenstufen ähnliche Aufbau der Schädeldecke ist bereits im Bilddatensatz vorhanden und nicht durch die Segmentierung entstanden.

Bewertung

Prototyp «Alpha»	
Geschwindigkeit	00:13 min
Grösse des STL-Exports	31'654 KB
Detailtreue des Modells	2
Oberflächenglätte des Modells	5

Prototyp «Beta»	
Geschwindigkeit	40:55 min*
Grösse des STL-Exports	679'021 KB
Detailtreue des Modells	6
Oberflächenglätte des Modells	4

Freie Software 3D Slicer	
Geschwindigkeit	00:12 min
Grösse des STL-Exports	23'962 KB
Detailtreue des Modells	2
Oberflächenglätte des Modells	3

Industrie-Software VGSTUDIO MAX	
Geschwindigkeit	27:47 min
Grösse des STL-Exports	124'681 KB
Detailtreue des Modells	5
Oberflächenglätte des Modells	3

*Da der verwendete Rechner einen relativ kleinen Arbeitsspeicher aufweist, wurde der Prototyp «Beta» an einem leistungsstärkeren Rechner getestet. Dieser Rechner weist folgende Spezifikationen auf:

Prozessor: Intel® Xeon® CPU E5-1650 v4 @ 3.60GHz 3.60GHz
RAM: 128.00 GB

Daher ist die angegebene Zeit nicht mit den anderen Zeiten zu vergleichen.

3.3.4 Fazit

Durch die Validierung konnten einige wichtige Erkenntnisse erlangt werden. So ist zum Beispiel zu beobachten, dass der Prototyp «Alpha» ähnliche Ergebnisse liefert wie die freie Software 3D Slicer. Ähnlich verhält es sich mit dem Prototyp «Beta» und der Industriesoftware VGSTUDIO MAX. Auch bei diesen beiden Programmen sind die Ergebnisse vergleichbar.

Betrachtet man die **Geschwindigkeit**, liegt der Prototyp «Alpha» und 3D Slicer klar im Vorteil. Innerhalb weniger Sekunden wurde das gewünschte Gewebe segmentiert und als STL-Datei exportiert. Ganz im Gegensatz zur Software VGSTUDIO MAX. Dort dauerte eine Segmentierung bis zu einer knappen halben Stunde. Noch länger dauerte eine Segmentierung mit dem Prototyp «Beta». Dort musste man sich bis zu 40 Minuten gedulden, um ein Ergebnis zu erhalten. Und dies war auch nur möglich, weil man einen leistungsstarken Rechner verwendete, da die Segmentierung mit diesem Prototyp einen grossen Teil des Arbeitsspeichers in Anspruch nimmt. Die Geschwindigkeit spiegelt sich auch in der **Dateigrösse** der STL-Datei wider, welche bei den schnelleren Softwaretools bedeutend kleiner ist.

Legt man den Fokus jedoch auf die **Detailtreue und Oberflächenglätte**, sind die Vor- und Nachteile anders verteilt. Besonders auffällig ist die Segmentierung aus dem Testfall 1 – «Thorax». Betrachtet man dort das Ergebnis der Nierensegmentierung des Prototyps «Alpha» in Abbildung 33, fällt die **Detailtreue** besonders negativ auf. Die feinen Nierenvenen und -arterien, welche bei der Niere hinein und hinaus führen, werden unnatürlich dargestellt und entsprechen nicht der Realität. Bei den anderen drei Programmen ist die Detailtreue in diesem Beispiel gut bis hervorragend zu beurteilen. Ein weiteres Beispiel für eine schlechte Detailtreue ist im Testfall 3 – «Cranium» zu sehen. In Abbildung 39 ist die Schädeldecke abgebildet. Beim Prototyp «Alpha» und bei der Software 3D Slicer ist die Knochennaht – die sogenannte Sutur – nicht mal ansatzweise zu erkennen. Ganz im Gegensatz zum Prototyp «Beta» und der Industriesoftware VGSTUDIO MAX, wo die Sutur deutlich abgebildet ist.

Die **Oberflächenglätte** lässt sich besonders gut durch den Testfall 2 – «Truncus» beurteilen. Beim Prototyp «Alpha» und der Software 3D Slicer ist ein schichtartiger Aufbau zu erkennen, wobei dieser beim Prototyp «Alpha» mittels Smoothing noch etwas geglättet wird. Trotzdem sind auch dort die Schichten deutlich zu erkennen. Diese Schichten sind bei VGSTUDIO MAX nur bei genauerer Betrachtung zu sehen. Ausserdem sind sie dort viel feiner vorzufinden. Von den Schichten ist im Prototyp «Beta» nichts zu erkennen. Dies ist deshalb erstaunlich, weil bei diesem Segmentierungswerkzeug kein Smoothing stattgefunden hat. Anders sieht es bei der Segmentierung des Schädels aus. Im Testfall 3 – «Cranium» erkennt man in der Abbildung 38 die Schichten sowohl beim Prototyp «Beta» als auch – noch etwas ausgeprägter – bei der Industrie-Software VGSTUDIO MAX. Ganz im Gegensatz zu den anderen beiden Programmen, wobei der Prototyp «Alpha» besonders positiv auffällt.

Zusammenfassend lässt sich sagen, dass der Prototyp «Beta» zu den besten Ergebnissen führt, wenn man nur die Detailtreue und Oberflächenglätte beurteilt. Ein grosser Nachteil dieses Programms ist jedoch die hohe Beanspruchung des Arbeitsspeichers während der Segmentierung, bei welcher herkömmliche Rechner schnell an ihre Leistungsgrenze stossen. Dies hängt jedoch klar mit der erhöhten Auflösung zusammen, die mit dem Prototyp «Beta» erreicht werden kann.

4 Schlussfolgerung

4.1 Zusammenfassung der Ergebnisse

Im Rahmen dieser Arbeit wurden zunächst bestehende Segmentierungsapplikationen analysiert, um die Unterschiede hinsichtlich Qualität und Handhabung von freier Software zu Industrie-Software zu untersuchen. Dazu wurde bei allen Programmen beispielhaft die Segmentierung der rechten Niere durchgeführt. Das Ergebnis zeigt, dass es mit allen Applikationen gelungen ist, die Segmentierung zu vollziehen. Man kann aber auch diverse Unterschiede feststellen, sowohl im angewendeten Segmentierungsverfahren als auch im Ergebnis selbst. So ist es besonders die Detailtreue, die in der Industrie-Software im Vergleich zu den freien Applikationen besonders positiv auffällt. Erstaunlich sind die Ergebnisse auch im Hinblick auf die Oberflächenglätte. Dort übertrifft die eine freie Software und übertrifft damit sogar die Industrie-Software.

Die Erkenntnisse aus der Analyse der Segmentierungsapplikationen flossen direkt bei der Entwicklung des eigenen Segmentierungsprototyps mit ein. So wurde der Oberflächenglätte und Detailtreue besondere Aufmerksamkeit geschenkt. Es wurden zwei verschiedene Prototypen implementiert, wobei man sich jeweils in unterschiedlichen Software-Bibliotheken bediente. Beide Prototypen verwenden zur Segmentierung das Schwellenwertverfahren, also ein pixelorientiertes Segmentierungsverfahren, bei welchem bei jedem einzelnen Voxel über die Zugehörigkeit zur jeweiligen Region entschieden wird, ohne seine Nachbarelemente zu betrachten. Zur Optimierung der Oberflächenglätte kommt beim ersten Prototyp «Alpha» ein Glättungs-Algorithmus zum Einsatz. Der zweite Prototyp «Beta» führt die Segmentierung mit einer deutlich höheren Auflösung durch, wobei die Oberfläche automatisch glatter wirkt und das Model zusätzlich eine viel höhere Detailtreue aufweist. Die Auflösung lässt sich zudem manuell einstellen, um so den Bedürfnissen des Benutzers gerecht zu werden.

Beide Prototypen wurden überprüft und validiert, wobei sie an unterschiedlichen Datensätzen getestet und mit bereits bestehenden Segmentierungsapplikationen verglichen wurden. Die Validierung zeigt, dass besonders der Prototyp «Beta» mit der einstellbaren Auflösung überzeugende Ergebnisse liefert. So erzielt er die beste Detailtreue unter den getesteten Programmen, wobei auch eine Industrie-Software getestet wurde. Auch die Oberflächenglätte überzeugt, wenn man bedenkt, dass bei diesem Prototyp kein Glättungs-Algorithmus zum Einsatz kommt. Insgesamt kann dieser Prototyp als gute Alternative zu Industrie-Software betrachtet werden. Ganz im Gegensatz zum Prototyp «Alpha», welcher zwar eine gute Oberfläche aufweist, die Detailtreue jedoch zu wünschen übrig lässt. Feine Strukturen sind nicht zu erkennen und der Glättungs-Algorithmus führt besonders bei kleineren Strukturen zu unnatürlichen Formen. Der Vorteil dieses Prototyps ist allerdings seine Geschwindigkeit, womit er eher als Alternative für freie Software angesehen werden kann.

4.2 Zukünftiger Forschungsbedarf

Wie man bereits in der Einleitung erfahren konnte, gibt es unterschiedliche Segmentierungsverfahren, welche je nach Einsatzgebiet zu favorisieren sind. Bei beiden implementierten Prototypen wird das pixelorientierte Schwellenwertverfahren verwendet, welches den Nachteil hat, dass nicht zusammenhängende Strukturen entstehen können. Um diesen Missstand zu verhindern, könnte man den Segmentierungs-Algorithmus in den Prototypen so ändern, dass ein regionenorientiertes Segmentierungsverfahren angewendet wird. Ein solches Verfahren wurde auch in Kapitel 2 bei der Analyse von bereits bestehenden Segmentierungsapplikationen untersucht und führte zu guten Ergebnissen. Der Vorteil dieses Verfahren ist, dass nur jenes Gewebe segmentiert wird, welches vom Benutzer ausgewählt wurde.

Zusätzlich ist auch der Rechenaufwand und der damit verbundene erhöhte Zeitaufwand des Prototyps «Beta» eine wichtige negative Eigenschaft, die es weiter zu untersuchen gilt. Hierbei gäbe es andere Alternativen zum verwendeten Segmentierungs-Algorithmus, die man ausprobieren könnte. Wichtig dabei ist jedoch, dass die Qualität der Segmentierung auf demselben Niveau bleibt, da diese zum jetzigen Zeitpunkt auf einem sehr guten Stand ist.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Bachelor-Thesis unterstützt und motiviert haben und so massgeblich zum Gelingen der Arbeit beigetragen haben.

Besonderen Dank möchte ich meinem Betreuer und Dozent Prof. Dr. Philipp Schütz aussprechen, der mich während dieser Arbeit begleitet und stets tatkräftigt unterstützt hat. Seine Fachkompetenz und seine Fähigkeit, komplexe Sachverhalte auf verständliche Art zu erklären, haben mich bei jeglichen Problemen sehr weitergeholfen. Ausserdem war er sich auch zu später Stunde nie zu schade, um auf ein E-Mail zu antworten. Das wusste ich sehr zu schätzen.

Vielen herzlichen Dank!

Literaturverzeichnis

- 3D Slicer. (o.D.). *3D Slicer*. Abgerufen am 26. 04. 2019 von <https://www.slicer.org/>
- Deserno, T. M. (2011). Medizinische Bildverarbeitung. In R. Kramme, *Medizintechnik* (S. 825-846). Berlin/Heidelberg: Springer-Verlag.
- Dua, S., Kandiraju, N., & Chowirappa, P. (10. 10. 2009). *Region Quad-Tree Decomposition Based Edge Detection for Medical Images*. Abgerufen am 03. 05. 2019 von <https://benthamopen.com/FULLTEXT/TOMINFOJ-4-50>
- Handels, H. (2009). *Medizinische Bildverarbeitung: Bildanalyse, Mustererkennung und Visualisierung für die computergestützte ärztliche Diagnostik und Therapie*. Leipzig: Vieweg+Teubner.
- Hebb, A., & Poliakov, A. (2009). Imaging Of Deep Brain Stimulation Leads Using Extended Hounsfield Unit CT. *Stereotactic and Functional Neurosurgery* 87.3, S. 155-160.
- ITK-SNAP. (27. 04. 2018). *ITK-SNAP*. Abgerufen am 26. 04. 2019 von <http://www.itksnap.org/pmwiki/pmwiki.php?n=Main.HomePage>
- Jähne, B. (2012). *Digitale Bildverarbeitung*. Berlin/Heidelberg: Springer-Verlag.
- Kramme, R. (2017). *Medizintechnik; Verfahren - Systeme - Informationsverarbeitung*. Berlin/Heidelberg: Springer Verlag.
- Lehmann, T., Oberschelp, W., Pelikan, E., & Repges, R. (1997). *Bildverarbeitung für die Medizin*. Berlin/Heidelberg: Springer-Verlag.
- Steinbrecher, R. (1993). *Bildverarbeitung in der Praxis*. München und Wien: Oldenbourg.
- Tarjetas. (o.D.). *Kisspng*. Abgerufen am 05. 03. 2019 von <https://de.kisspng.com/png-n7gz3a/>
- Wikipedia. (14. 05. 2018). *Vgstudio*. Abgerufen am 26. 04. 2019 von <https://de.wikipedia.org/wiki/Vgstudio>
- Wintermantel, E., & Ha, S. W. (2009). *Medizintechnik; Life Science Engineering*. Berlin Heidelberg: Springer Verlag.

Anhang

In diesem Kapitel werden die Quelltexte der im Rahmen dieser Arbeit implementierten und getesteten Prototypen «Alpha» und «Beta» dargestellt.

Prototyp «Alpha»

```
# -*- coding: utf-8 -*-
"""
Author: Adrian von Wyl
BAT: 3D-Segmentierer basierend auf freier Software, FS19
Prototyp "Alpha"
"""

#Quelle:
#https://pyscience.wordpress.com/2014/09/11/surface-extraction-creating-a-mesh-from-pixel-data-using-
#https://nbviewer.jupyter.org/urls/bitbucket.org/somada141/pyscience/raw/master/20140908_SurfaceExtraction/

#%%
import vtk
from vtk.util import numpy_support
import os
import numpy
import plotly
from plotly.graph_objs import *
plotly.plotly.sign_in("somada141", "1t2qb5b9y1")

#%%
def vtkImageToNumPy(image, pixelDims):
    pointData = image.GetPointData()
    arrayData = pointData.GetArray(0)
    ArrayDicom = numpy_support.vtk_to_numpy(arrayData)
    ArrayDicom = ArrayDicom.reshape(pixelDims, order='F')

    return ArrayDicom

def plotHeatmap(array, name="plot"):
    data = Data([
        Heatmap(
            z=array#,
            #scl='Greys'
        )
    ])
    layout = Layout(
        autosize=False,
        title=name
    )
    fig = Figure(data=data, layout=layout)

    return plotly.plotly.iplot(fig, filename=name)

#%%
import vtk
from IPython.display import Image
def vtk_show(renderer, width=400, height=300):
    """
    Takes vtkRenderer instance and returns an IPython Image with the rendering.
    """
    renderWindow = vtk.vtkRenderWindow()
    renderWindow.SetOffScreenRendering(1)
    renderWindow.AddRenderer(renderer)
    renderWindow.SetSize(width, height)
    renderWindow.Render()

    windowToImageFilter = vtk.vtkWindowToImageFilter()
    windowToImageFilter.SetInput(renderWindow)
    windowToImageFilter.Update()

    writer = vtk.vtkPNGWriter()
    writer.SetWriteToMemory(1)
    writer.SetInputConnection(windowToImageFilter.GetOutputPort())
    writer.Write()
    #data = str(buffer(writer.GetResult()))

    return Image(0) #Image(data)

#Quelle: https://programtalk.com/python-examples/vtk.vtkSmoothPolyDataFilter/
def ApplySmoothFilter(polydata, iterations, relaxation_factor):
    """
    Smooth given vtkPolyData surface, based on iteration and relaxation_factor.
    """
```

```

    smoother = vtk.vtkSmoothPolyDataFilter()
    smoother.SetInputData(polydata)
    smoother.SetNumberOfIterations(iterations)
    smoother.SetFeatureAngle(80)
    smoother.SetRelaxationFactor(relaxation_factor)
    smoother.FeatureEdgeSmoothingOn()
    smoother.BoundarySmoothingOn()
    # smoother.GetOutput()##ReleaseDataFlagOn()
    smoother.AddObserver("ProgressEvent", lambda obj, evt:
        UpdateProgress(smoother, "Smoothing surface..."))

    return smoother.GetOutputPort()

PathDicom = r"INSERT INPUT FOLDER PATH HERE"
reader = vtk.vtkDICOMImageReader()
reader.SetDirectoryName(PathDicom)
reader.Update()

# Load dimensions using `GetDataExtent`
_extent = reader.GetDataExtent()
ConstPixelDims = [_extent[1]-_extent[0]+1, _extent[3]-_extent[2]+1, _extent[5]-_extent[4]+1]

# Load spacing values
ConstPixelSpacing = reader.GetPixelSpacing()

#shiftScale = vtk.vtkImageShiftScale()
#shiftScale.SetScale(reader.GetRescaleSlope())
#shiftScale.SetShift(reader.GetRescaleOffset())
#shiftScale.SetInputConnection(reader.GetOutputPort())
#shiftScale.Update()

# In the next cell you would simply get the output with 'GetOutput' from 'shiftScale' instead of 'reader'

ArrayDicom = vtkImageToNumPy(reader.GetOutput(), ConstPixelDims)
plotHeatmap(numpy.rot90(ArrayDicom[:, 128, :]), name="CT_Original")

threshold = vtk.vtkImageThreshold()
threshold.SetInputConnection(reader.GetOutputPort())
threshold.ThresholdByLower(113) #set threshold here
threshold.ReplaceInOn()
threshold.SetInValue(0) # set all values below threshold to 0
threshold.ReplaceOutOn()
threshold.SetOutValue(1) # set all values above threshold to 1
threshold.Update()

ArrayDicom = vtkImageToNumPy(threshold.GetOutput(), ConstPixelDims)
plotHeatmap(numpy.rot90(ArrayDicom[:, 128, :]), name="CT_Thresholded")

#time
dmc = vtk.vtkDiscreteMarchingCubes()
dmc.SetInputConnection(threshold.GetOutputPort())
dmc.GenerateValues(1, 1, 1)
dmc.Update()

fufu=ApplySmoothFilter(dmc.GetOutput(),15,0.3) #set iterations and smoothing factor here
mapper = vtk.vtkPolyDataMapper()
#mapper.SetInputConnection(dmc.GetOutputPort())
mapper.SetInputConnection(fufu)

actor = vtk.vtkActor()
actor.SetMapper(mapper)

renderer = vtk.vtkRenderer()
renderer.AddActor(actor)
renderer.SetBackground(1.0, 1.0, 1.0)

camera = renderer.MakeCamera()
camera.SetPosition(-500.0, 245.5, 122.0)
camera.SetFocalPoint(301.0, 245.5, 122.0)
camera.SetViewAngle(30.0)
camera.SetRoll(-90.0)
renderer.SetActiveCamera(camera)
vtk_show(renderer, 600, 600)

camera = renderer.GetActiveCamera()
camera.SetPosition(301.0, 1045.0, 122.0)
camera.SetFocalPoint(301.0, 245.5, 122.0)
camera.SetViewAngle(30.0)
camera.SetRoll(0.0)
renderer.SetActiveCamera(camera)
vtk_show(renderer, 600, 600)

writer = vtk.vtkSTLWriter()

```

```
#writer.SetInputConnection(dmc.GetOutputPort())
writer.SetInputConnection(fufu)
writer.SetFileTypeToBinary()
writer.SetFileName("new segmentation alpha.stl")
writer.Write()

renderer_window = vtk.vtkRenderWindow()
renderer_window.AddRenderer(renderer)
renderer_interactor = vtk.vtkRenderWindowInteractor()
renderer_interactor.SetRenderWindow(renderer_window)
renderer.SetBackground(1,1,1)
renderer_window.SetSize(1024, 768)

renderer_interactor.Initialize()
renderer_window.Render()
renderer_interactor.Start()
```

Prototyp «Beta»

```
# -*- coding: utf-8 -*-
"""
Author: Adrian von Wyl
BAT: 3D-Segmentierer basierend auf freier Software, FS19
Prototyp "Beta"
"""

#Source:
#https://www.kaggle.com/akh64bit/full-preprocessing-tutorial
#https://www.raddq.com/dicom-processing-segmentation-visualization-in-python/
#alternative to %matplotlib
#Source: https://stackoverflow.com/questions/35595766/matplotlib-line-magic-causes-syntaxerror-in-python-script
from IPython import get_ipython
get_ipython().run_line_magic('matplotlib', 'inline')
#When output is to be displayed in separate window: switch 'inline' to 'qt'

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import dicom
import os
import scipy.ndimage
import matplotlib.pyplot as plt

from skimage import measure, morphology
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
from glob import glob

# Some constants
INPUT_FOLDER = r'INSERT INPUT FOLDER PATH HERE'
output_path = working_path = r'INSERT OUTPUT FOLDER PATH HERE'
patients = os.listdir(INPUT_FOLDER)
patients.sort()
g = glob(INPUT_FOLDER + '/*.dcm')

# Print out the first 5 file names to verify we're in the right folder.
print ("Total of %d DICOM images.\nFirst 5 filenames:" % len(g))
print ('\n'.join(g[:5]))

# Load the scans in given folder path
def load_scan(path):
    slices = [dicom.read_file(path + '/' + s) for s in os.listdir(path)]
    slices.sort(key = lambda x: int(x.InstanceNumber))
    try:
        slice_thickness = np.abs(slices[0].ImagePositionPatient[2] - slices[1].ImagePositionPatient[2])
    except:
        slice_thickness = np.abs(slices[0].SliceLocation - slices[1].SliceLocation)
    for s in slices:
        s.SliceThickness = slice_thickness

    return slices

def get_pixels_hu(scans):
    image = np.stack([s.pixel_array for s in scans])
    # Convert to int16 (from sometimes int16),
    # should be possible as values should always be low enough (<32k)
    image = image.astype(np.int16)

    # Set outside-of-scan pixels to 0
    # The intercept is usually -1024, so air is approximately 0
    image[image == -2000] = 0

    # Convert to Hounsfield units (HU)
    intercept = scans[0].RescaleIntercept
    slope = scans[0].RescaleSlope

    if slope != 1:
        image = slope * image.astype(np.float64)
        image = image.astype(np.int16)

    image += np.int16(intercept)

    return np.array(image, dtype=np.int16)

id=0
patient = load_scan(INPUT_FOLDER)
imgs = get_pixels_hu(patient)

np.save(output_path + "fullimages_%d.npy" % (id), imgs)

file_used=output_path+"fullimages_%d.npy" % id
imgs_to_process = np.load(file_used).astype(np.float64)

plt.hist(imgs_to_process.flatten(), bins=100, color='c')
```

```

plt.xlabel("Hounsfield Units (HU)")
plt.ylabel("Frequency")
plt.show()

imgs_to_process = np.load(output_path+'fullimages_{}.npy'.format(id))

def sample_stack(stack, rows=5, cols=5, start_with=0, show_every=2):
    fig,ax = plt.subplots(rows,cols,figsize=[12,12])
    for i in range(rows*cols):
        ind = start_with + i*show_every
        ax[int(i/rows),int(i % rows)].set_title('slice %d' % ind)
        ax[int(i/rows),int(i % rows)].imshow(stack[ind],cmap='gray')
        ax[int(i/rows),int(i % rows)].axis('off')
    plt.show()
sample_stack(imgs_to_process)

print ("Slice Thickness: %f" % patient[0].SliceThickness)
print ("Pixel Spacing (row, col): (%f, %f) " % (patient[0].PixelSpacing[0], patient[0].PixelSpacing[1]))

id = 0
imgs_to_process = np.load(output_path+'fullimages_{}.npy'.format(id))

def resample(image, scan, new_spacing=[1,1,1]):
    # Determine current pixel spacing
    spacing = map(float, ([scan[0].SliceThickness] + scan[0].PixelSpacing))
    spacing = np.array(list(spacing))

    resize_factor = spacing / new_spacing
    new_real_shape = image.shape * resize_factor
    new_shape = np.round(new_real_shape)
    real_resize_factor = new_shape / image.shape
    new_spacing = spacing / real_resize_factor

    image = scipy.ndimage.interpolation.zoom(image, real_resize_factor)

    return image, new_spacing

print ("Shape before resampling\t", imgs_to_process.shape)
imgs_after_resamp, spacing = resample(imgs_to_process, patient, [1,1,1]) #set resolution here
print ("Shape after resampling\t", imgs_after_resamp.shape)

def plot_3d_stl(image, threshold=-300):

    # Position the scan upright,
    # so the head of the patient would be at the top facing the camera
    p = image.transpose(2,1,0)
    # p = p[:,::-1] #mirror model: -1

    from stl import mesh

    #Source: https://pypi.org/project/numpy-stl/ (Creating Mesh objects from a list of vertices and faces)
    verts, faces = measure.marching_cubes_classic(p, threshold)
    segmentation = mesh.Mesh(np.zeros(faces.shape[0], dtype=mesh.Mesh.dtype))
    for i, f in enumerate(faces):
        for j in range(3):
            segmentation.vectors[i][j] = verts[f[j],:]

    #Write the mesh to file "new segmentation beta"
    segmentation.save('new segmentation beta.stl')

    fig = plt.figure(figsize=(10, 10))
    ax = fig.add_subplot(111, projection='3d')

    # Fancy indexing: `verts[faces]` to generate a collection of triangles
    mesh = Poly3DCollection(verts[faces], alpha=0.1)
    face_color = [0.5, 0.5, 1]
    mesh.set_facecolor(face_color)
    ax.add_collection3d(mesh)

    ax.set_xlim(0, p.shape[0])
    ax.set_ylim(0, p.shape[1])
    ax.set_zlim(0, p.shape[2])

    plt.show()

plot_3d_stl(imgs_after_resamp, 113) #set threshold here

```